CENA 10.000 zł
ISSN 0867-3918 INDEKS 377112

64 PLUS 4 & AMIGA

3/'92

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



Drodzy Czytelnicy!

Mamy nadzieję, że będzie odpowiadać Wam wprowadzona w - tym numerze zmiana kroju i wielkości czcionek. Dzięki temu zabiegowi nieco „sztucznie" uzyskaliśmy więcej miejsca na odrobinę grafiki i materiały merytoryczne.

Informujemy również, że nasze pismo można w dalszym ciągu **zaprenumerować** - co daje pewność systematycznego otrzymywania (drogą pocztową). Nasz miesięcznik kosztuje w prenumeracie 10.000 zł. Prenumeratę można zawrzeć na okres nie krótszy niż dwa miesiące, w dowolnym okresie, maksymalnie do końca roku kalendarzowego. Wykupujący prenumeratę nie ponoszą kosztów przesyłki pocztowej.

REDAKCJA

P.S. Prosimy Pana Macieja Walczaka z Gniezna o podanie swojego adresu, którego brak uniemożliwia nam realizację zamówienia.

Wadliwa dystrybucja „64 plus 4 & Amiga" przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

zapraszamy wszystkich chętnych
do prowadzenia kolportażu
„64 plus 4 & Amiga"
(kluby, studia i sklepy komputerowe, księgarnie, osoby indywidualne itd.) do współpracy!

Oferujemy korzystne warunki!

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.



miesięcznik nr 3(17)
marzec 1992
cena 1 egz.: 10.000
64 PLUS 4

WYDAWCA: ABUK Spółka z o.o.
REDAGUJĄ: Waldemar Szczygieł (redaktor naczelny) z zespołem.
ADRES REDAKCJI: Redakcja „64 plus 4", 85-166 Bydgoszcz 43, skrytka pocztowa 64.
OKŁADKA: Piotr Bartz.
SKŁAD: ABUK
DRUK: W.Z.G. Wąbrzeźno, okładka: Z.P. POLRASTER, Bydgoszcz.

OD REDAKCJI

W numerze :

Od redakcji 3
Upominki, upominki . .4
Zegar 5
Kto pyta nie błądzi 6
Przerwania 7
Spis zestawu PDP na C-64 (nr 14) 8
Sprite'y i Basic 10
Assembler 6510 - lekcja 8 11
Auto-fire 14
Public Domain Pack . .15
Reklama 17
Zdrowie i komputer . .19
Manhuter NY 20
TRACKDISK 21
Kącik początkującego kodera 24
PDP na Amigę zestaw nr 14 25
ARP library - cz. 5 27
Power Packer 28
Kurs języka C 28

DLA WSZYSTKICH

Upominki, upominki

Przedstawiamy listę prenumeratorów naszego pisma, którzy wylosowali nagrody i upominki (zapowiadaliśmy je w numerach 11 i 12/91 oraz w numerze 1/92):

Amigę CDTV wylosował pan Antoni Margro z Poznania, joysticki wylosowali: Mateusz Czubak - Łosice, Piotr Witosławski - Łódź, Adam Półkośnik - Białystok, Adam Sęk - Zator, Aleksander Waszczuk - Olsztyn, Krzysztof Latkowski - Wrocław, Janusz Czaporowski - Skierniewice, Mariusz Wrzesiński - Szczecin, Klara Morawska - Międzyborów, Leszek Bakalarczyk - Bydgoszcz, Szczepan Gala - Sosnowiec, Tadeusz Kułaga - Chełm, Artur Karaźniewicz - Kraszewo, Jerzy Koenig - Legnica, Andrzej Brodzik - Drohiczyn, Andrzej Malinowski - Nowa Wieś, Katarzyna Niemiec - Myszków, Piotr Paruzel - Tarnowskie Góry, Tomasz Damps - Gdańsk, Rafał Doniec - Tarnowskie Góry.

Wśród wszystkich, którzy zamówili komplet taśm PDP na C-64 rozlosowaliśmy - zgodnie z obietnicą - 10 programów Voicetracker V4.0T. Program ten otrzymują:

Konstanty Falkowski - Tychy, Piotr Stolc - Gdynia, Grzegorz Nasternak - Żyrardów, Rafał Łysy - Barcin, Andrzej Szura - Gdańsk, Dariusz Skupień - Ochojec, Artur Cybułka - Opole, Karol Gemza - Siedlec, R. Tarasiewicz - Białowieża, M. i R. Raksimowicz - Gdynia.

Program Voicetracker w wersji dyskowej wylosowali:

Tomasz Dąbrowski - Elbląg, Dariusz Żywalewski - Rzepin, Władysław Justyński - Rzezawa, Jacek Chmielewski - Kołobrzeg, Sławomir Buchcik - Kłodzko, Zygmunt Figura - Gdynia, Witold Raj - Łódź, Marian Skrzecz - Warszawa, Marcin Woszczyński - Osięciny, Jacek Wasilewski - Pruszków.

Wyniki losowania programu D-Mon Professional publikujemy na str. 26.

Listy stu osób, które wylosowały upominki - wybaczcie nam - nie będziemy publikować (zważywszy na małą objętość naszego pisma) - zajęła by ona sporo miejsca.

Wszystkie nagrody i upominki prześlemy pocztą.

Wszystkim serdecznie gratulujemy!

OGŁOSZENIA

Zamienię nowy C-64 z gwarancją, joysticki, Black Box IV, magnetofon, instrukcja na A-500 z dopłatą. Agnieszka Sowińska, Kielce, tel. :558-71.

Sprzedam drukarkę do Commodore 64 MPS 803 - nowa. Bydgoszcz, tel. 42-17-12.

AMIGA 500/plus/2000 - najlepsze gry oraz programy użytkowe, super nowości - wysyłka pocztą. Ekspresowe terminy, katalogi gratis. „SOFTSTUDIO", Tysiąclecia 54/6, 31-610 Kraków, tel. (012) 485150.

Zamienię ATARI 800 XE = kartridge i gry na kasetach na Commodore C-64 II lub C-64 I. Gałązka Marian, ul. Akacjowa 13/25, 41-200 Sosnowiec.

Wymienię programy C-64 kaseta. Michał Groth, Pomorska 22A/20, 80-333 Gdańsk.

KLUB UŻYTKOWNIKÓW AMIGI zaprasza. Oferujemy porady, tanie dyskietki, wolny dostęp do klubowych programów. Olech Tomasz, Al. Piłsudzkiego 30/19, 41-303 Dąbrowa Górnicza.

Nawiążę kontakt z posiadaczami Commodore 16,116, plus4. Krzysztof Mróz, ul. M. Maliny 2a/23, 41-200 Sosnowiec.

Sprzedam C128D, monitor, mysz, dyski. Ł. Pychyński, 46-320 Praszka-Kowale, Wieluńska 7a.

AMIGA 500, COMMODORE 64 - wymiana programów. Nowości. Koperta + znaczek. Paweł Witek, Karłowicza 45/55, 58-506 Jelenia Góra.

Sprzedam C-64II + podstawowy osprzęt za 2 mln. P. Cymbor, Żory, tel. 344-504.

Sprzedam Amigę 500 na gwarancji, 1MB, 100 dysków, modulator TV, 3 joysticki. Szczecin, te. 791-608.

Amiga - gry, użytki, problemy, pomoc, katalog gratis. Ded, 41-300 Dąbrowa Górnicza, 1 Maja 4/4.

Commodore 64, magnetofon, joystick, oprogramowanie sprzedam. Engelbrecht Grzegorz, ul. Rejtana 9/50, 84-200 Wejherowo.

Kupię programy do drukowania polskich liter na C-64 z MPS -1230. L. Cepil, Oś. Sikorskiego 14/45, 28-100 Busko Zdrój.

KOMPUTEROWA FIRMA USŁUGOWA „TREND"
COMMODORE AMIGA 500 - 3000
LITERATURA W J. POLSKIM(!) I OPROGRAMOWANIE
INFORMACJA: DYSKIETKA LUB KOPERTA+ZNACZEK
KONTAKT: Rafał Wierzbicki, ul. Budziszyńska 112/28, 54-436 Wrocław.

Korespondencyjny Klub Użytkowników Amigi.
Informacja = koperta + znaczek. Marcin Bohdziul, ul. Koszarowa 18/12, Szczecin - Dąbie.

Commodore 64 - programy (taśma), literaturę - wymienię; koperta+znaczek. Andrzej Orechwa, ul. Ogrodowa 11/123, 00-893 Warszawa.

ZEGAR

C-16

Program „ZEGAR" wykorzystuje mało znaną metodę programowania grafiki:

współrzędnych punktu nie podaje się wprost, np.:

DRAW 1, 10, 25 (X=10, Y=25)

lecz względnie:

jako przesunięcie wobec ostatniego położenia kursora graficznego (PC-Pixel Cursor).

Dla niezorientowanych podaję, że kursor graficzny jest niewidoczny na ekranie i znajduje się w ostatnio rysowanym (lub magazynowanym) punkcie; można go także umieszczać w dowolnym miejscu komendą

LOCATE X, Y;

gdzie X, Y są współrzędnymi, które ma przyjąć PC.

A teraz przykłady:

- Narysujmy na ekranie punkt komendą DRAW 1, 160, 100. Mamy narysować punkt oddalony od poprzedniego o 20 w prawo i o 30 jednostek w górę (oznacza to, że do wsp. X mamy dodać 20, a od wsp. Y odjąć 30):

  DRAW 1, +20, -30

- Jeszcze większe zastosowanie ma rozkaz umożliwiający rysowanie punktu oddalonego o daną odległość pod zadanym kątem od PC. Ten rozkaz:

  DRAW 1, 40; 45

spowoduje zapalenie punktu odległego o 40 jednostek pod kątem 45 stopni.

W prosty sposób można zaprogramować ruch wskazówki umocowanej w punkcie (160, 100):

```
10 GRAPHIC 1, 1
20 FOR KAT=0 TO 360 STEP 10
30 DRAW 1, 160, 100 TO 50;KAT
40 DRAW 0, 160, 100 TO 50;KAT
50 NEXT KAT
60 GOTO 20
```

A oto program rysujący spiralę:

```
10 GRAPHIC 1, 1
20 LOCATE 160, 100:R=1
30 DRAW TO R; R*30
40 R=R+1
50 GOTO 30
```

Przejdźmy teraz do opisu trudniejszych fragmentów programu ZEGAR.

Linia 140 jest odpowiedzialna za wyświetlanie ekranu tekstowego podczas przygotowywania rysunku zegara (ekran graficzny zobaczymy dopiero po wykonaniu linii 400).

Stała SK wyznaczona w linii 160 to kąt, o jaki wskazówka godzinowa przesuwa się w ciągu sekundy (SK=12*60*60).

W liniach 320 - 390 wyliczane są początkowe położenia wskazówek.

Program wydziela ze zmiennej systemowej TI$ trzy zmienne: MI, HO i SE (oznaczające odpowiednio ilość minut, godzin i sekund), a następnie przelicza je na odpowiednie kąty ustawienia wskazówek.

Wzór z linii 360 jest konsekwencją płynnego (bez skoków) ruchu wskazówki godzinowej.

Właściwy program jest zawarty w liniach 430 - 560. Jest on wykonywany dokładnie co sekundę dzięki pętli w linii 430 sprawdzającej, czy ilość sekund sprzed poprzedniego wywołania programu się zmieniła. Ta część jest odpowiedzialna za przesuwanie (o ile to jest konieczne) wskazówek.

Dorobienie budzika - alarmu pozostawiam chętnym.

Życzę miłej zabawy.

Wojciech Kazimierczak

```
100 REM          ZEGAR
110 REM NAPISAL WOJCIECH KAZIMIERCZAK
120 REM
130 COLOR 0,1:COLOR 4,1:COLOR 1,7
140 GRAPHIC 1,1:GRAPHIC 0,1
150 PRINT"CZEKAJ..."
160 SK=360/43200
170 REM ---RYS. ZEGARA---
180 CIRCLE 1,160,100,90
190 CIRCLE 1,160,100,94:PAINT 1,160,7
200 FOR I=30 TO 360 STEP 30
210 FOR J=1 TO 4
220 CIRCLE 1,160,100,94+J, ,I-2,I+2
230 NEXT J
240 NEXT I
250 CIRCLE 1,160,100,99
260 REM --USTAWIANIE CZASU--
270 PRINT "AKTUALNY CZAS:";TI$
280 PRINT "WPISZ NOWY (GGMMSS) LUB WCISNIJ
    RETURN."
290 INPUT T$
300 IF T$ <>" " THEN TI$=T$
310 REM---USTAWIANIE KATOW WSKAZOWEK--
320 MI=VAL(MID$(TI$,3,2)):M=MI*6
330 DRAW 1,160,100,TO 85;M
340 HO=VAL(LEFT$(TI$,2))
350 IF HO>12 THEN HO=HO-12
360 H=HO*30+MI*.5
370 DRAW 1,160,100 TO 50;H
380 SE=VAL(RIGHT$(TI$,2))
390 S=SE*6:DRAW 1,160,100 TO 90;S
400 GRAPHIC 1,0:T$=RIGHT$(TI$,2)
410 REM --GLOWNA PETLA--
420 REM -WYWOLANIE CO 1S.-
430 IF T$=RIGHT$(TI$,2) THEN 430
440 T$=RIGHT$(TI$,2)
450 DRAW 0,160,100 TO 88;S:S=S+6
460 DRAW 1,160,100 TO 88;S
470 IF S=360 THEN S=0:ELSE 490
480 DRAW 0,160,100 TO 85;M:M=M+6
490 DRAW 1,160,100 TO 85;M
500 H=H+SK:HC=INT(H)
510 IF HC=H1 THEN 540
520 DRAW 0,160,100 TO 50;H-SK
530 IF HC=360 THEN H=0
540 DRAW 1,160,100 TO 50;H
550 H1=HC
560 GOTO 430
```



KTO PYTA NIE BŁĄDZI

Do naszej redakcji napływa sporo listów z różnymi pytaniami - na niektóre z nich postaramy się dziś odpowiedzieć.

PYTANIE

Porównuję klawisz RESTORE C-64 z moim C-16. Na C-64 można na przykład przerwać program maszynowy i zdefiniować stan wyjściowy. Czy mogę dodatkowo wmontować do C-16 klawisz RESTORE?

ODPOWIEDŹ

Klawisz RESTORE w C-64 działa bezpośrednio na wejście NMI procesora 6510. Niestety procesor C-16 i Plus/4 nie mają linii NMI. Znaczy to, że dodatkowo nie ma możliwości wmontowania klawisza RESTORE. Aby w tych komputerach zdefiniować stan wejściowy mamy następującą możliwość: naciskamy klawisz RUN/STOP równocześnie z klawiszem RESET znajdującego się obok wyłącznika, C-16 „zamelduje się" wbudowanym językiem maszynowym Tedmon. Jeśli teraz naciśniemy „X", to znajdziemy się znów w trybie wejściowym. Uprzednio wprowadzony program nie zostanie skasowany!

PYTANIE

Jeśli nagram programy z C-64 na taśmę, to nie mogę ich później załadować na C-16 i odwrotnie. Dlaczego się tak dzieje i czy można temu coś zaradzić?

ODPOWIEDŹ

C-64 i C-16 używają dla współpracy z magnetofonem różnych formatów nagrywania. Załadowanie programów z C-64 do C-16 w bezpośredni sposób nie jest zatem możliwe.

PYTANIE

Mam kilka programów które nie funkcjonują w moim C-16 z rozszerzeniem 64kB. Czy moja dodatkowa pamięć jest uszkodzona?

ODPOWIEDŹ

Nie, rozszerzenie pamięci nie jest uszkodzone. System operacyjny C-16 dokonuje innego - w przypadku większej ilości RAM'u - podziału pamięci (w szczególności jeśli używamy grafiki wysokiej rozdzielczości). W rezultacie nie pracują programy w kodzie maszynowym. Są dwie możliwości aby temu zaradzić: albo wmontować wyłącznik w celu odłączenia rozszerzenia (skomplikowane), albo spróbować zasymulować wersję 16KB instrukcjami Basic'a. Najczęściej pomaga ostatni sposób. Instrukcja ta wygląda następująco:

```
POKE 1331,246:POKE 1332,63:SYS32768
```

Rozkazy muszą zostać wprowadzone w trybie bezpośrednim przed każdorazowym załadowaniem programu. Wasz C-16 lub Plus/4 zamelduje się teraz zwykłą planszą tytułową informując że liczba wolnych bajtów wynosi 12277 (jak standardowo przy C-16 bez rozszerzenia).

PYTANIE

Zasilacz mojego C-16 (z rozszerzeniem 64KB) jest (przede wszystkim podczas używania magnetofonu) tak ciepły, że powoduje zakłócenia w pracy komputera i muszę go na moment wyłączać. W czym leży tego przyczyna?

ODPOWIEDŹ

To zjawisko występuje przede wszystkim wtedy gdy zasilacz jest przeciążony. Układy rozszerzenia pamięci komputera oraz magnetofon(pobiera on bardzo dużo prądu) zasilane są z jednego zasilacza. W niektórych wypadkach zasilacz jest tak przeciążony, że dochodzi do przepalenia bezpieczników. Stwierdziliśmy że zasilacz innego komputera - SPECTRUM można bez problemów podłączyć do C-16 (nie pasuje do Plus/4). Ma tą samą wtyczkę a co najważniejsze większą wydajność prądową. Jeśli tylko zdobędziemy go gdzieś na giełdzie komputerowej to spróbujcie go podłączyć. Konstrukcja zasilacza do C-16 szczegółowo była omawiana w numerze „64 Plus 4" z listopada 1990 r.

PYTANIE

Czy są programy, które automatycznie tłumaczą programy z C-16 lub Plus/4 na C-64?

ODPOWIEDŹ

Takiego programu nie ma. Programy w Basic'u, w których nie występują instrukcje PEEK, POKE lub SYS można jednakże uruchamiać na obu komputerach.

PYTANIE

Czy mogę joystick z C-64 podłączyć również do C-16 lub Plus/4?

ODPOWIEDŹ

W zasadzie na to pytanie muszę odpowiedzieć - NIE. Jednak przy odrobinie żyłki majsterkowicza jest to możliwe.

PYTANIE

Mam drukarkę wykorzystującą złącze Centronics. Jak będzie ona współpracować z moim C-16 lub Plus/4?

ODPOWIEDŹ

Użytkownicy C-64 mają duże możliwości podłączania drukarki do komputera. Pierwsza - przez kabel połączeniowy między User-Port C-64 a drukarką.

Rozwiązanie te (zwane również Software-Interface) z powodu braku User-Port w C-16 jest praktycznie nie do zrealizowania. Druga - przez tzw. Centronics-Interface. Jest to połączenie między portem szeregowym (serial) komputera, a wejściem Centronics drukarki. Port ten w C-16 jest identyczny z C-64.

RG

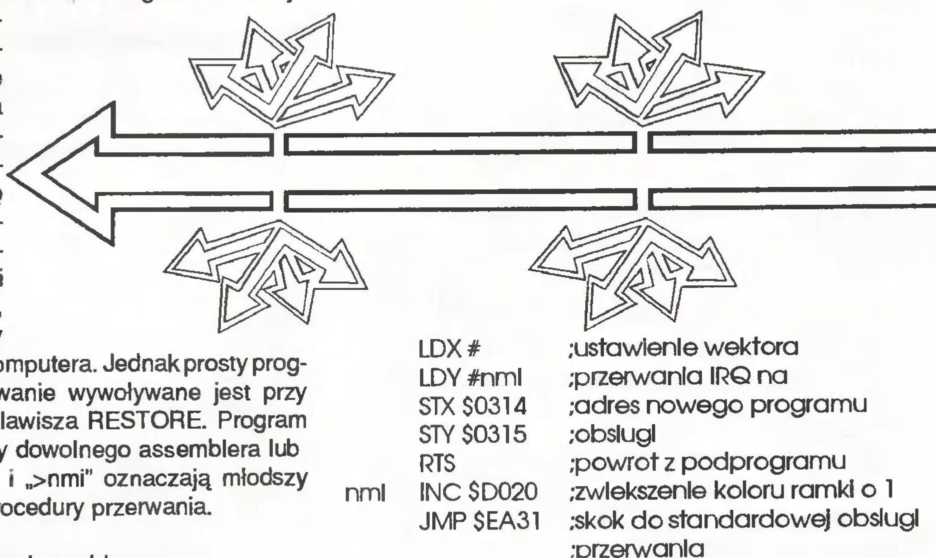
Przerwania w każdym komputerze spełniają wiele istotnych funkcji. Dzięki nim komputer może przeglądać w odpowiednich odstępach czasu klawiaturę podczas działania dowolnego programu, również dzięki przerwaniom możemy tworzyć wiele wspaniałych efektów graficznych. Także każde naciśnięcie klawisza RESTORE wywołuje specjalny rodzaj przerwania.

PRZERWANIA

Ogólnie rzecz biorąc działanie przerwań polega na tym, że procesor - podczas wykonywania programu - otrzymuje w pewnym momencie sygnał z zewnątrz, który nakazuje mu wykonanie specjalnego programu wskazanego przez odpowiednie komórki pamięci. Po jego wykonaniu procesor powraca z przerwania do poprzedniej pracy, a więc kontynuuje wykonywanie przed chwilą przerwanego programu.

Działanie przerwań zorganizowane zostało w ten sposób, że stan procesora po wykonaniu programu przerwania jest dokładnie taki sam, jak przed jego wykonaniem, tak więc procesor jakby nie zauważa swojego dodatkowego działania.

W C64 wyróżniamy zasadniczo dwa główne rodzaje przerwań. Pierwsze, to przerwania zwane NMI. Przerwanie takie jest wywoływane przez naciśnięcie klawisza RESTORE, który jest podłączony bezpośrednio do samego procesora. Mogłoby się wydawać, że wywołanie przerwania zachodzi tylko w przypadku użycia kombinacji RUN-STOP i RESTORE, gdyż wtedy zauważamy wyraźne zmiany na ekranie komputera. Jednak prosty program pokaże nam, że przerwanie wywoływane jest przy każdorazowym naciśnięciu klawisza RESTORE. Program możemy wpisać przy pomocy dowolnego assemblera lub monitora. Przy czym „<nmi" i „>nmi" oznaczają młodszy i starszy bajt adresu nowej procedury przerwania.



```
     LDX #<nmi    ;ustawienie wektora
     LDY #>nmi    ;przerwania NMI na
     STX $0318    ;adres nowego programu
     STY $0319    ;obslugi
     RTS          ;powrot z podprogramu
nmi  INC $D020    ;zwiekszenie koloru ramki o 1
     RTI          ;powrot z przerwania
```

Jak widać adres, do którego procesor ma skakać po wywołaniu przerwania jest zapamiętany w dwóch komórkach pamięci o adresach $0318 oraz $0319 (dziesiętnie 792 oraz 793). Dzięki zmianie wektora przerwania NMI możemy w prosty sposób zabezpieczać programy w języku maszynowym przed zatrzymaniem przez kogoś niepowołanego, a po zablokowaniu działania klawisza RUN-STOP można zabezpieczać również programy w BASIC'u.

Drugi rodzaj przerwań nazywa się IRQ. Są one wywoływane przez wewnętrzne układy komputera pięćdziesiąt razy w ciągu jednej sekundy. Standartowo przerwanie to wywołuje układ CIA, zajmujący się obsługą urządzeń zewnętrznych. Są to tak zwane „przerwania klawiatury", gdyż właśnie dzięki nim komputer pięćdziesiąt razy na sekundę sprawdza czy został naciśnięty klawisz.

Wektor przerwania IRQ znajduje się w komórkach $0314/$0315 (788/789). Działanie przerwań można wyraźnie zobaczyć wpisując prosty program w assemblerze:

```
     LDX #        ;ustawienie wektora
     LDY #nmi     ;przerwania IRQ na
     STX $0314    ;adres nowego programu
     STY $0315    ;obslugi
     RTS          ;powrot z podprogramu
nmi  INC $D020    ;zwiekszenie koloru ramki o 1
     JMP $EA31    ;skok do standardowej obslugi
                  ;przerwania
```

Skok do standardowej obsługi przerwania musi zostać wykonany, gdy chcemy aby została zachowana obsługa klawiatury, a więc jest również niezbędny, aby nasza procedura przerwania mogła współpracować z programem w BASIC'u.

Można jednak wywoływać przerwania IRQ również za pomocą procesora graficznego VIC. Są to tak zwane „przerwania graficzne". To właśnie dzięki nim można otrzymać na ekranie telewizora wiele ciekawych efektów, począwszy od kolorowych pasów, czy dzielenia ekranu na część tekstową i graficzną, a skończywszy na zaawansowanych efektach (na przykład otwieranie ramek).

C-64

Wywołanie tego przerwania jest nieco trudniejsze. Po całkowitym zablokowaniu przerwań należy zakazać układowi CIA, ich generacji, a układowi VIC rozkazać, aby dawał sygnał procesorowi.

Warto jeszcze wiedzieć, iż przerwania graficzne są wywoływane zawsze w momencie, gdy wiązka elektronów wyświetlająca obraz osiągnie linię o numerze zapisanym w komórce $D012. Ze względu na to, że ekran ma więcej niż 256 linii najstarszy bit wywołania zapisany jest w najstarszym bicie komórki $D011. Natomiast odczyt tych komórek informuje o aktualnej pozycji wiązki elektronów.

Mając taki zasób wiadomości możemy pokusić się o napisanie prostej procedury dzielącej ekran na dwie części. Jedną, wyświetlaną przy użyciu pierwszego generatora znaków i drugą według drugiego generatora:

```
        SEI          ;Wylaczenie przerwan.
        LDX #<lrq1   ;Zmiana wektora IRQ na
        LDY #<irq1   ;adres pierwszej procedury
        STX $0314    ;obslugi.
        STY $0315    ;
        LDA #$7F     ;Zatrzymanie przerwan
        STA $DC0D    ;generowanych przez CIA.
        LDA #$01     ;Uruchomienie generowania
        STA $D01A    ;przerwan przez VIC.
        LDA #$1B     ;Przerwanie bedzie wywolane
        STA $D011    ;gdy wiazka elektronow
        LDA #$32     ;osiagnie linie numer $32
        STA $D012    ;(poczatek ekranu)
        CLI          ;Wlaczenie przerwan.
        RTS          ;Powrot.
Irq1    INC $D019    ;Przyjecie przerwania VIC'a.
        LDA #$15     ;Wlaczenie pierwszego
        STA $D018    ;generatora znakow.
        LDA #$92     ;Przerwanie zostanie wywolane
        STA $D012    ;w polowie ekranu.
        LDX #$<lrq2  ;Ustawienie wektora IRQ na
        LDY #$>lrq2  ;adres drugiej procedury.
        STX $0314    ;
        STY $0315    ;
        JMP $EA81    ;Wyjscie z przerwania bez
                     ;obslugi klawiatury.
Irq2    INC $D019    ;Przyjecie przerwania VIC'a.
        LDA #$17     ;Wlaczenie drugiego
        STA $D018    ;generatora znakow.
        LDA #$32     ;Przerwania zostanie wywolane
        STA $D012    ;na gorze ekranu.
        LDX #<lrq1   ;Ustawienie wektora IRQ na
        LDY #>lrq2   ;adres pierwszej procedury.
        STX $0314    ;
        STY $0315    ;
        JMP $EA31    ;Skok do obslugi klawiatury.
```

Napewno wszyscy zauważyli, że po wykonaniu pierwszej procedury nie skaczemy do obsługi klawiatury. Dlaczego? Po prostu druga procedura jest wywoływana zanim wiązka elektronów osiągnie koniec ekranu, czyli po niepełnej 1/50 sekundy. Nie ma potrzeby tak częstego wywoływania obsługi klawiatury, więc stosujemy skok do szybkiego powrotu z przerwania.

Gdy zdeasemblujemy kawałek ROM'u od adresu $EA81 to zobaczymy sekwencję rozkazów zdejmującą ze stosu stan wszystkich rejestrów. Należy zawsze pamiętać o wykonaniu tej operacji, gdyż w momencie wywołania przerwania wszystkie rejestry są automatycznie zapamiętywane na stosie.

Oczywiście nie trzeba zawsze odczytywać klawiatury na końcu przerwania. Gdy nie ma takiej potrzeby wystarczy zamiast $EA31 wpisać skok pod $EA7E.

Mam nadzieję, że dzieki poznanym informacjom będziecie mogli wzbogacić swoje programy w BASIC'u, jak i w assemblerze o nowe, ciekawe efekty.

JARRI

SPIS ZESTAWU PUBLIC DOMAIN PACK

NR 14 (LUTY '92)

Jak zawsze, tak i na lutowym Public Domain Pack'u umieściliśmy kilka programów demonstracyjnych. Pierwszy z nich to najnowsza produkcja jednej z najlepszych polskich grup na C64 - Parados. Jest to kolekcja obrazków nowego grafika grupy o pseudonimie Picasso. Obrazki zostały wykonane w trybach multicolor oraz FLI.

Drugim programem demonstracyjnym jest dwuczęściowe demko stworzone w całości przez jedną osobę. Maduplec, bo tak brzmi pseudonim autora, wykonał do tego dema zarówno cały kod, jak i oprawę dźwiękową i graficzną.

Oprócz programów demonstracyjnych umieściliśmy również najnowsze numery dwóch Polskich magazynów dyskowych. O jednym z nich, AXEL NEWS, już wspominaliśmy na łamach naszego pisma, jednak drugi, zatytułowany CIABAH nie gościł jeszcze na naszych PDP. Magazyn ten robiony jest przez szczecińską grupę Crazy Boys przy wsparciu kolegów z grupy Skylight. Niestety, zarówno rozwiązanie programu obsługującego ten magazyn, jak i same teksty pozostawiają wiele do życzenia. Z pewnością warto by pomyśleć o nowym kodzie i nowej grafice.



Oprócz tych kilku programów typowo rozrywkowych umieściliśmy również kilka ciekawych tzw. „użytków"

1) **LYNX XVI+**
Jest to program pomocniczy służący do obsługi dysku. Po uruchomieniu na ekranie ukazuje się menu zawierające następujące opcje:
- create - zatwierdzone przez użytkownika pliki program łączy w jeden plik o długości równej sumie połączonych plików. Nazwę tego pliku podajemy z klawiatury, nazwy plików, z których został on utworzony są kasowane z katalogu.
- dissolve - plik utworzony przez opcję create zostaje rozpakowany do pierwotnej postaci, a - wcześniej - połączone pliki ponownie ukazują się w katalogu.
- header - opcja ta umożliwia podejrzenie, jakie pliki są ukryte w spakowanym (opcją create) pliku bez potrzeby rozpakowania go.
- copier i backup - opcje te nie są jeszcze umieszczone w tej wersji programu.
- edit - dzięki tej opcji możemy w prosty sposób manipulować katalogiem dyskietki. Możemy wstawiać pomiędzy nazwy programów sepatory, dowolnie zmieniać typ programów oraz wykonywać szereg innych operacji na katalogu.
- reader - powoduje odczytanie pliku i wydrukowanie go na ekranie w formie znaków ASCII.
- sector - prosty edytor sektorów dyskietki.

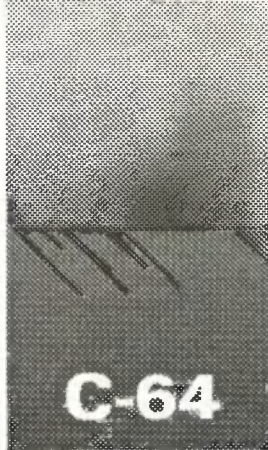
Ze względu na bezpośrednią ingerencję programu w dane zapisane na dyskietce nie polecam początkującym zabawy na dyskach zawierających istotne dane.

2) **SIDEBORDER LOGO EDITOR V1.0**
Jest to ostatnia wersja programu służącego do tworzenia grafiki trzykolorowej. Jego atrakcyjność polega jednak w głównej mierze na tym, że można rysować również na bocznej ramce! Niestety wyświetlenie tego rodzaju obrazków nie jest łatwe i obecnie w Polsce potrafi tego dokonać tylko nieliczna grupa osób. Wkrótce jednak na łamach naszego pisma postaram się omówić sposób otwierania bocznych ramek w C64.

3) **INTELPAINT**
Jest to program graficzny posiadający jedynie podstawowe opcje, jak rysowanie obrazka po punkcie, rysowanie linii i okręgu oraz wypełnianie obszarów. Oprócz tego program umożliwia rysowanie na dowolnym powiększonym fragmencie obrazu. Oryginalnym rozwiązaniem jest wprowadzenie specjalnego trybu graficznego o nazwie „interlace". Dzięki niemu można tworzyć obrazki w rozdzielczoci 320 na 200 piskeli przy użyciu trzech kolorów atramentu - w każdym polu osiem na osiem pikseli.

4) **FLI EDITOR V3.2**
Ten program jest w zasadzie połączeniem dwóch edytorów graficznych przystosowanych do pracy w różnych technikach. Podobny dla obu programów jest sposób zorganizowania „warsztatu" użytkownika. Pierwszy z nich umożliwia tworzenie grafiki w trybie FLI, drugi natomiast w zwykłym trybie wielokolorowym.

5) **4*4 CHARMAKER**
Program ten służy do tworzenia własnego kroju czcionki o wielkości 32 na 32 piksele. Tworzona czcionka jest zapamiętywana w czterech generatorach znaków. W każdym z nich zapamiętany jest jeden poziomy rządek litery. Sam program jest dosyć prosty. Oferuje jedynie najpotrzebniejsze narzędzia, jak na przykład zmiana koloru czy skasowanie aktualnie rysowanej litery.



6) **F(R)ONT EDITOR 3**
Podobnie jak poprzedni program tak i ten jest edytorem czcionek. Do tworzenia znaków wykorzystuje on jednak jeden generator. Tak więc można tworzyć zestawy znaków o rozmiarach 8 na 8, 16 na 16 oraz 256 na 64 piksele, zarówno w trybie wysokiej rozdzielczości, jak i wielokolorowym. Program posiada wiele przydatnych opcji.

7) **THE GRAFIX-PACK II**
Bardzo użyteczny zestaw narzędzi do zamiany formatów obrazków pomiędzy najpopularniejszymi programami. Oprócz tego można również fragmenty obrazków przenosić na sprite'y lub na generator znaków. Prosty w obsłudze - opiera się na starannie wykonanym systemie okienek.

Jarosław "JARRI" Horodecki

Przypominamy zawartość zestawu nr 13 (styczeń '92):

- **Char Zoomer V3.1.**
- **Colour Bar Editor V3.0.**
- **Hires + A-FLI Editor.**
- **FLI Designer V1.1.**
- **Dirmaster V3.1**
- **Accesinus.**
- **Music Routine Cruncher V1.5.**
- **Gandalf Protector V3.0.**
- **Gandalf Coder V1.0.**
- **Disk - tape copy V1.0.**
- **Gnd-packer V1.0.**

O zasadach nabywania programów Public Domain piszemy na str. 16.

C-64

SPRITE'Y i BASIC

Każdy posiadacz komputera C-64 niejednokrotnie oglądał programy demonstacyjne lub gry wykorzystujące tzw. sprite'y. Ich użycie jest bowiem najwygodniejszym sposobem tworzenia obiektów, które mają poruszać się niezależnie od tła. W tym artykule postaram się przybliżyć problem wykorzystywania tych obiektów w programach pisanych w języku BASIC.

Wyświetlaniem sprite'ów na ekranie zajmuje się w C-64 układ VIC. Aby mógł je wyświetlić należy jednak podać mu kilka istotnych danych oraz zdefiniować kształt obiektu. Każdy sprite w C-64 ma wielkość 24 na 21 pikseli, z czego można łatwo obliczyć, iż na zapisanie jego kształtu potrzebne są 63 bajty.

Jednak, jak i gdzie te dane umieścić? Po włączeniu komputera układ VIC, odpowiedzialny za wyświetlanie obrazu w C-64, wykorzystuje do swoich celów pierwszy bank pamięci (od adresu 0 do 16383) i tylko z niego może on pobierać dane o kształcie sprite'ów. Jeżeli dane o nich chcemy umieścić w innym obszarze pamięci musimy przełączyć bank, z którego korzysta VIC, czego jednak nie polecam początkującym użytkownikom. Adres, od którego wpisujemy dane można znaleźć korzystając z prostego wzoru: **adr = 64 * nr_spr**, gdzie nr_spr jest numerem kolejnym zbioru danych opisujących sprite'a w aktualnie włączonnym banku.

Ponieważ jeden bank ma 16384 bajty więc najwyższym numerem sprite'a może być 255. Nie znaczy to jednak, że C-64 może wywietlać na ekranie 256 (255 i numer 0) sprite'ów równocześnie, ale to, że teoretycznie możliwe jest zapamiętanie 256 wzorów sprite'ów, które mogą być wykorzystane na przykład do animacji.

W ostatnim zdaniu celowo użyłem słowa „teoretycznie", a to dlatego, że nie możemy używać obszaru pamięci od adresu 0 do 2048, gdyż jest on wykorzystany przez system (wektory skoków, stos, pamięć ekranu) oraz od adresu 4096 do 8192, gdyż w tym miejscu VIC „widzi" standardowe generatory znaków. Ponieważ od adresu 2048 znajdują się dane o programie w basic'u, więc najwygodniej jest umieszczać wzory naszych obiektów od adresu 8192 do 16384, co daje nam dla programu w basic'u około 6kB pamięci.

Przejdźmy teraz do projektowania naszych obiektów. Najwygodniej jest dokonać tego przy pomocy dowolnego edytora sprite'ów, gdyż wpisywanie danych ręcznie jest raczej uciążliwe. Dane o każdym obiekcie są zapamiętane kolejnymi poziomymi rzędami. Ponieważ w poziomie, jak już wcześniej wspomniałem, sprite ma 24 piksele, każdy poziomy rządek zapisany jest w trzech bajtach. Rządków tych jest 21, więc dane zajmują 63 bajty, co wynika z prostych obliczeń. W pamięci dane o każdym obiekcie są ułożone kolejnymi rządkami. Dla naszych potrzeb zdefiniujmy sobie sprite'a w kształcie prostokąta, wypełniając cały obszar danych wartościami 255 (czyli 11111111 w systemie dwójkowym).

Wykona to za nas pierwsza linia programu:

```
10 FOR I=0 TO 63: POKE (8192+I), 255: NEXT
```

Jak widać nasze dane znajdują się pod adresem 8192, więc są to dane o numerze 128.

Teraz należy powiedzieć VIC'owi, skąd ma pobierać dane o każdym ze sprite'ów. Informacje te są umieszczone zawsze zaraz za pamięcią ekranu, standardowo od adresu 2040, a ponieważ C-64 na ekranie może równocześnie wyświetlić tylko 8 sprite'ów, dane te kończą się pod adresem 2047. My jednak na początek poruszymy tylko pierwszego sprite'a, więc informacje o jego kształcie wpisujemy pod adres 2040.

W naszym przypadku wpiszemy tam wartość 128.

```
20 POKE 2040,128
```

Po wykonaniu tych czynności musimy zapoznać się z jeszcze kilkoma rejestrami obsługującymi sprite'y. Obszar rejestrów procesora VIC rozpoczyna się od adresu 53248, a kończy na adresie 53294.

W tym obszarze znajdują się rejestry odpowiedzialne za działanie VIC'a, a więc także i te które obsługują nasze sprite'y. Najważniejszy jest rejestr o adresie 53296. Jego osiem bitów odpowiada za włączenie lub wyłączenie każdego z ośmiu sprite'ów. Tak więc gdy wstawimy tam wartość 1,zostanie włączony tylko pierwszy sprite (bo jest to 00000001 dwójkowo), natomiast gdy wpiszemy wartość 15 zostaną włączone sprite'y pierwszy,drugi, trzeci oraz czwarty. Proste, prawda?

Równie ważne są rejestry, w których zapamiętujemy pozycję poziomą i pionową każdego sprite'a. Są one umieszczone na przemian od adresu 53248. Czyli pozycję poziomą pierwszego sprite'a wpisujemy pod adres 53248, pionową 53249, poziomą drugiego 53250, pionową 53251 itd. Wydaje mi się, że to również jest raczej proste. Może nawet trochę za proste.

Uważny czytelnik zapewne zauważy, iż pozycja pozioma jest zapisana tylko w jednobajtowym rejestrze, a nasz Commodore w najwyższej rozdzielczości może wyświetlić 320 pikseli w każdej linii. Tak więc korzystając z jednobajtowego rejestru możemy wyświetlić sprite'a najwyżej w trzech-czwartych ekranu. Aby wyświetlić

go w drugiej części ekranu należy skorzystać z kolejnego rejestru - 53264.

Podobnie jak w rejestrze mówiącym o aktywności sprite'ów, tak i tu każdy bit odpowiada jednemu sprite'owi. Jeżeli dany bit jest ustawiony, to do pozycji poziomej danego sprite'a dodawane jest 256.

To tyle podstawowych informacji.

Teraz przedstawię kilka przykładów.

Dopiszmy do naszego programu następujące trzy linie:

```
30 POKE 53269,1
40 POKE 53248,24: POKE 53249,5050 POKE 53264,0
```

Linia 10 naszego programu to oczywiście włączenie pierwszego sprite'a, linia 40 to wyświetlenie go na pozycji (24;100), a linia 50 informuje VIC'a, że wybieramy lewą część ekranu.

Tylko dlaczego - mimo, że współrzędne nie są zerowe-sprite pojawił się w lewym, górnym rogu ekranu? Dzieje się tak dlatego, że wyświetlanie sprite'ów jest możliwe również na ramce, tak więc punkt o wpółrzędnych (0;0) znajduje się na samej górze ekranu z lewej strony. W tej pozycji sprite jest niewidoczny. Dopiero przy zastosowaniu specjalnych technik, osiągalnych tylko z poziomu języka maszynowego, możliwe jest zllikwidowanie ramek i wtedy sprite ten również staje się widoczny.

Teraz proponuję wstawić w linii 50 zamiast zera jedynkę i ponownie uruchomić program. Sprite ukaże się w prawej części ekranu na pozycji 256+24=280.

Zanim przystąpimy do dalszej pracy proponuję poeksperymentować trochę z wartościami w liniach 40 oraz 50.

W linii 50 ma sens zmiana wartości tylko pomiędzy 1 i 0 (gdyż obsługujemy tylko jednego sprite'a).

Gdy już opanowaliśmy technikę poruszania sprite'a możemy pokusić się o stworzenie programiku, który będzie za nas poruszał go po ekranie. Najlepiej niech będzie to coś w rodzaju piłeczki.

Na początku musimy zapoznać się z możliwymi maksymalnymi pozycjami w poziomie i w pionie naszego sprite'a, czyli takimi, przy których nie będzie „wychodził" poza ekran. Kilka eksperymentów z naszym programikiem i już mamy: minimum w poziomie: 24, maksimum: 64 i 1 w 53264 (czyli 320), minimum w pionie: 50, maksimum w pionie: 228. Będąc w posiadaniu tych informacji możemy pokusić się o napisanie programu poruszającego naszą prostokątną „piłeczkę".

Linie 10, 20 oraz 30 pozostaną takie same. Zmiany zaczynamy od lini 40.

```
40 X=100: Y=100
45 A=1: B=1
50 X=X+A: Y=Y+B
60 IF X=24 OR X=320 THEN A=-A
65 IF Y=50 OR Y=228 THEN B=-B
70 POKE 53249,Y
80 POKE 53248,X-INT(X/256)*256
85 POKE 53264,X/256
90 GOTO 50
```

Jeszcze krótki opis: w linii 40 ustalamy początkowe współrzędne dla sprite'a, w linii 45 ustalamy współczynniki dodania do pozycji sprite'a. Linia 50 to zmiana starych współrzędnych na nowe. Linie 60 i 65 to sprawdzenie warunków wyjścia sprite'ów poza ekran, jeżeli współrzędna sprite'a spełnia jeden z postawionych warunków sprite zaczyna się przesuwać w przeciwnym kierunku. Linie 70 do 85 to wpisywanie współrzędnych sprite'a do odpowiednich rejestrów. Przy czym uwzględniamy możliwość przechodzenia sprite'a pomiędzy prawą i lewą częścią ekranu. Linia 90 to: zapętlenie programu.

Niestety po uruchomieniu programu natychmiast zauważamy podstawową wadę pisania programów w basic'u. Oczywiście chodzi o szybkość wykonywania programu. Jak widać nie ma większego sensu z poziomu tego języka obsługiwać większej ilości sprite'ów, gdyż - ze względu na szybkość wykonywania - nie wyglądałoby to ciekawie. Mimo to radzę pokusić się o napisanie prostej gry zręcznościowej w basic'u wykorzystującej sprite'y. W tym celu musimy się jednak zapoznać z kolejną dawką teorii, ale o tym za miesiąc.

Jarosław "JARRI" Horodecki

ASSEMBLER 6510

- LEKCJA 8

Witam w kolejnym odcinku naszego kursu assemblera. Dzisiaj dokończymy listę rozkazów procesora 6510 oraz zajmiemy się arytmetyką dziesiętną (BCD)

Zacznijmy od omówienia pozostałych rozkazów:

■ BIT - test bits (testowanie bitów)

N	V	D	I	Z	C
*	*	-	-	*	-

rozkaz	kod	cykle
BIT $nn	24	3
BIT $nnnn	2C	4

Rozkaz ten wykonuje operację logiczną AND na akumulatorze i podanym rejestrze, jednak wynik nie jest nigdzie zapisywany, natomiast zmieniane są odpowiednie znaczniki. I tak znacznik Z jest ustawiony gdy wynikiem operacji jest zero, natomiast w znaczniki N i V wpisywane są kolejno: bit 7 oraz bit 6 podanej komórki pamięci.

C-64

NOP - no operation (bez operacji)

N	V	D	I	Z	C
-	-	-	-	-	-

rozkaz	kod	cykle
NOP	EA	2

Wbrew pozorom rozkaz ten jest dość często używany przez programistów. Może on być z powodzeniem stosowany przy tworzeniu pętli czasowych, opóźnień potrzebnych do uzykania wielu skomplikowanych efektów graficznych, kiedy to praca procesora musi być idealnie zgrana z wyświetlanym obrazem. Rozkaz NOP jest też często używany do kasowania sekwencji niepotrzebnych rozkazów (na przykład po znalezieniu „nieśmiertelności" w grze).

SEI - set interrupt disable bit (zabronienie przyjmowania przerwań)

N	V	D	I	Z	C
-	-	-	1	-	-

rozkaz	kod	cykle
SEI	78	2

Wykonanie tego rozkazu powoduje wyłączenie przez procesor systemu przerwań. Reaguje on tylko na przerwania wywołane rozkazem BRK lub wykonanie zimnego albo gorącego startu. Najczęściej rozkaz ten jest używany przy zmianie wektora przerwań na inny adres.

CLI - clear interrupt disable bit (pozwolenie na przyjmowanie przerwań)

N	V	D	I	Z	C
-	-	-	0	-	-

rozkaz	kod	cykle
CLI	58	2

Rozkaz ten powoduje włączenie systemu przerwań procesora. Od momentu jego wykonania procesor reaguje na wszystkie rodzaje przerwań. Rozkaz ten jest używany zwykle w zestawieniu z rozkazem SEI, jako włączenie przerwań po zmianie wektora.

BRK - break (przerwanie)

N	V	D	I	Z	C
-	-	-	1	-	-

rozkaz	kod	cykle
BRK	00	7

Rozkaz BRK stosowany jest główne jako tzw. „pułapki" w programach, które testujemy. Jego wykonanie powoduje wywołanie przerwania, którego wektor znajduje się pod adresem $0316/$0317 i wskazanie procedury inicjacji systemu takie, jak po naciśnięciu STOP+RESTORE. Wektor ten jest zawsze zmieniany przez monitory języka maszynowego na ich własny wektor gorącego startu.

RTI - return from interrupt (powrót z przerwania)

N	V	D	I	Z	C
pobrane ze stosu					

rozkaz	kod	cykle
RTI	40	6

Jest to rozkaz w pewien sposób podobny do RTS z tym, że zamiast z podprogramu wykonuje powrót z przerwania. RTI powoduje zdjęcie ze stosu rejestru znaczników oraz PC (licznika rozkazów). Używany jest zawsze na końcu każdej procedury obsługi przerwania.

SED - set decimal (włączenie trybu dziesiętnego)

N	V	D	I	Z	C
-	-	1	-	-	-

rozkaz	kod	cykle
SED	F8	2

CLD - clear decimal (wyłączenie trybu dziesiętnego)

N	V	D	I	Z	C
-	-	0	-	-	-

rozkaz	kod	cykle
CLD	D8	2

Dwa ostatnie rozkazy służą do włączania i wyłączania trybu dziesiętnego procesora. Tryb ten jest najczęściej używany przy wykonywaniu obliczeń zmiennopozycyjnych. Wykorzystywany również przy liczeniu punktów w grach, albowiem nie wymaga specjalnych procedur przeliczają-

cych liczby szesnastkowe na dziesiętne i gwarantuje łatwość przenoszenia danych na ekran.

Ale na czym właściwie polega arytmetyka dziesiętna? System ten został wymyślony, aby ułatwić obliczenia ludziom przyzwyczajonym do starych prawideł, na przykład, że 5+5=10, a nie $0A. Po prostu komputer zapomina o cyfrach powyżej 9. Każda cyfra jest zapamiętana w czterech bitach. I tak na przykład: 1 to 0001, a 9 to 1001. Wynika z tego, że w jednym bajcie możemy zapisać dwie cyfry, czyli każdą liczbę od 0 do 99. Jeżeli wynik dodawania przekracza ten zakres to ustawiany jest znacznik C (przeniesienia).

To tyle podstawowej teorii. Proponuję wpisać prosty przykład:

```
SED        ;Wlaczenie trybu dziesietnego.
LDA #$05   ;
CLC        ;
ADC #$05   ;Wykonanie dodawania 5+5.
BRK        ;Koniec.
```

Po powrocie do monitora w akumulatorze jest liczba $10, a ponieważ pracujemy w trybie dziesiętnym, więc odpowiada to po prostu liczbie 10. Jeżeli jednak suma dwóch liczb przekracza 99 - co się wtedy dzieje? Rozwiązanie tego problemu jest bardzo proste. Wystarczy po wykonaniu dodawania uwzględnić zawartość znacznika przeniesienia. Gdy wynik nie mieści się w jednym bajcie najwygodniej jest posłużyć się komórkami strony zerowej. Skorzystamy więc z komórek $FB oraz $FC. Oto procedura:

```
SED        ;Wlaczenie trybu dziesietnego.
LDA #$00   ;Wyzerowanie komorek, do ktorych
STA $FB    ;wpiszemy wynik.
STA $FC    ;
LDA #$55   ;Pierwszy skladnik do akumulatora.
CLC        ;
ADC #$55   ;Dodanie drugiego skladnika.
STA $FB    ;Odlozenie sumy do komorki $FB.
LDA $FC    ;Dodanie znacznika przeniesienia do
ADC #$00   ;komorki $FC i odlozenie wyniku do
STA $FC    ;tej samej komorki.
BRK        ;Koniec.
```

Po powrocie do monitora zajrzyjmy do komórek $FB oraz $FC. Znajdziemy tam następujące wartości: $01 oraz $10, czyli po prostu 110. Podobnie musimy postąpić w przypadku odejmowania z tym, że przed jego wykonaniem należy pamiętać o ustawieniu znacznika przeniesienia na 1, aby w przypadku niedomiaru komputer mógł pobrać brakujący bit. Jeżeli odejmowanie wykonujemy na liczbie dwubajtowej to po jego wykonaniu należy pamiętać o sprawdzeniu zawartości znacznika przeniesienia i w razie potrzeby odjąć 1 od starszego bajtu wyniku.

Mam nadzieję, że po tych kilku informacjach każdy z Was będzie w stanie bez problemu wykonać kilka prostych ćwiczeń:

1. Napisać program sumujący liczby od 0 do 99 korzystając z arytmetyki dziesiętnej.

2. Mając dane dwie dwubajtowe liczby zapisane w kodzie BCD wykonać na nich:
a) dodawanie
b) odejmowanie

3. Napisać program liczący od 0 do 9999. Program ma kolejne liczby wyświetlać na ekranie. Dodatkowo należy zrobić pętlę opóźniającą, aby kolejne liczby nie „przeleciały" zbyt szybko.

C-64

Na koniec obiecana miesiąc temu moja propozycja rozwiązania zadania z numeru styczniowego. Przypominam, że procedura miała przesłać dane z zadanego obszaru pamięci do drugiego obszaru o podanym adresie początku. Przyjmijmy, że adres początku tego obszaru zapamiętany jest w komórkach $FA/$FB, końca - $FC, $FD, a początku obszaru docelowego w $FE, $FF. Oto moja propozycja procedury:

```
      LDY #$00      ;Wyzerowanie rejestru Y.
petla LDA ($FA),Y   ;Poczatek petli.
      STA ($FE),Y   ;Przeladowaie danej.
      INC $FA       ;Zwiekszenie kolejnych
      INC $FE       ;komorek pamieci informujących
      BNE skok1     ;o adresie zrodlowym oraz
      INC $FB       ;docelowym.
      INC $FF       ;skok1
      LDA $FA       ;Sprawdzenie, czy adres poczatku
      CMP $FC       ;obszaru zrodlowego jest rowny
      BNE petla     ;adresowi konca tego obszaru.
      LDA $FB       ;Jezeli tak, to wyjscie z
      CMP $FD       ;procedury.
      BNE petla     ;
      RTS           ;Powrot z procedury.
```

Za miesiąc przedstawię propozycje rozwiązań dzisiejszych zadań oraz zapoznamy się z tzw. rozkazami niepublikowanymi procesora.

Jarosław "JARRI" Horodecki

AUTO FIRE!

Przy wielu grach niezbędne jest częste naciskanie przycisku fire w celu uzyskania wysokiego konta punktów. Czy może Wasz palec wskazujący przy niektórych grach jest trochę za wolny? Jeśli posiadacie joystick bez auto-fire to możecie go wyposażyć w tę funkcję samemu, a gry typu Fort Apocalipse czy Scramble pójdą nam jak po maśle.

W niniejszym artykule przedstawimy Wam auto-fire do samodzielnej budowy. Auto-fire to nic innego, jak prosty generator, który wysyła odpowiednie impulsy do komputera. Proponowany układ jest łatwy w montażu - wystarczy umieć posługiwać sie lutownicą. Szybkość sygnału fire ustawiamy potencjometrem. Do kontroli pracy układu służy dioda świecąca. Jeśli wyłączymy urządzenie to joystick będzie zachowywać się w pełni normalnie.

Aby uniknąć przebudowy komputera i joysticka, proponujemy zmontować całość w osobnym pudełku, jako „pośrednik" między posiadanym joystickiem a komputerem.

Sercem urządzenia jest timer NE555. Szybkość (częstotliwość) ustawiana jest przez układ RC składający się z kondensatora 0.1µF i potencjometru 25KΩ.

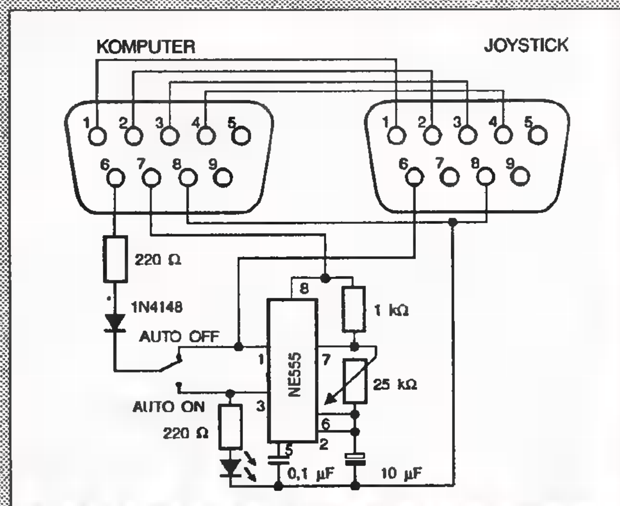
Dla bezpieczeństwa powinno się umieścić układ scalony w podstawce. Części montażowe zostały tak dobrane, aby nie było kłopotów z ich zakupem.

Montaż najlepiej jest wykonać na gotowej płytce z otworami - tzw. uniwersalnej.

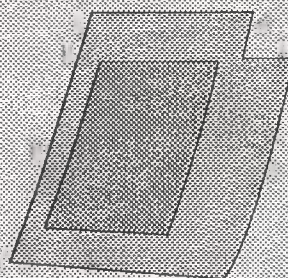
UWAGA:

Przy włączonym komputerze nie powinniśmy dołączać, ani rozłączać układu (dotyczy to wszystkich urządzeń peryferyjnych).





Schemat układu auto-fire.





PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C - 64

Styczeń '91 (nr 1)

STRONA A

- Mega demo grupy „VISION"-MIST2

STRONA B

- Preview do gry „UN SQUADRON"
- Preview do gry „PUZZLENOID"
- Preview do gry „TURRICAN"

Luty '91 (nr 2)

STRONA A

- TUNE OF MONTH
- LOGO WRITER V 2.0
- FAST CRUELCRUNCH
- WRATH+(DEMO)[02]
- DREPTACZ _ BASIC

STRONA B

- SWISS CHEESE/CFA
- DISK FAST LOADER

Marzec '91 (nr 3)

STRONA A

- FONT GRUB 1.0
- PROJEKTANT DUSZKÓW
- STRZAŁKA 64+
- PIRATEK - GRA
- V4.0 - SYMPHONIES
- CRUISER
- THE FIRST
- COMMERCIAL BREAK
- RELAKATOR 64
- KOREKTOR 64
- FLASH

STRONA B

- HOT SHOT nr9 (zach. magazyn fanów)
- BAD NEWS nr2 - j.w.
- DEMO - rekord - 290 SPRITE'ów!
- DEMO: NEW INTRO
- DEMO: LET'S DYCP
- KONTAKT CORNER _ adresy, kontakty
- NEW FAST - działa z 1541 I 1541 II
- CSLINKER V2.0

Kwiecień '91 (nr 4)

STRONA A

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku

STRONA B

- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika

Maj '91 (nr 5)

STRONA A

- CRUEL SOLIDERS - demo
- DESTINATION - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

STRONA B

- MEGA DEMO „INFOSYSTEM 91"

Czerwiec '91 (nr 6)

STRONA A

- FONTEDITOR
- SINDATA EDITOR
- COLOR EDITOR
- DISK - NOTER
- GWIAZDY - demo graficzne
- FILGRAEPH 2.2/BML
- NOTE TO FLI V 2.2
- AFLI - EDITOR V 1.2
- RESET - MON,8,1
- TURBO - ASS 5
- ...HIGHLIFE #5
- AXEL NEWS #1
- DISK NOTKA/PADUA

STRONA B

- PSC - MAG #9'06/91
- CONSPIRE?OREGON - demo
- CONTACT DEMO/ORE
- SHOWPIX

Lipiec '91 (nr 7)

STRONA A

- Mega demo „MY, OH MY!" grupy LIGHT

STRONA B

- Game Music Composer - edytor muzyczny grupy GRAFFITY Węgier.

Sierpień '91 (nr 8)

STRONA A

- MegaDemo „Unnamed" grupy CAMELOT
- Sound Killer - edytor muzyczny grupy TOPAZ
- AFLI - Editor graficzny techniki A-FLI
- Disk-Dos obsługa komend stacji dysków
- Noter v2.2 grupy TOPAZ
- IFFL - Squeezer kompresor dyskowy
- Dismaster+ - edytor do dyskietek
- Super Copy - DOS szybki program kopiujący do zbiorów

STRONA B

- Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues"

Wrzesień '91 (nr 9)

STRONA A:

- Mega Demo grupy FLASH

STRONA B:

- Hot Shot - magazyn dyskowy
- Code Sucker monitor - pr. użytkowy grupy PADUA
- Mountain Ride - gra w BASIC

Październik '91 (nr 10)

STRONA A i B:

- MEGA DEMO „AIRDANCE 4" grupy T.A.T.

Listopad '91 (nr 11)

STRONA A

- NEW LAW & ORDER!
- FLT/LEGOLAND
- FLT/LEGONOTE
- TERMINAT.2%/FLT

STRONA B

- SM. CRIMINAL #08
- SMALL BUT FINE
- HOLLY SMOKE/M12
- UNITE!/SYLVIO

Grudzień '91 (nr 12)

STRONA A

- ARMAGEDDON 5
- NOTE TO DEMO

STRONA B

- OUTRUN 2 MUS $ SFX
- AFTERBURNER/MON
- TRIVIA-GAME MUSIC
- FORM.I. SIMULATOR
- 2400AD END-TUNE
- NIGHTHUNTER DIGI
- ELIMINATOR MUSIC
- TOMCAT MUS./MON
- ZAMZARA TUNE/MON
- NOTE TO DISK
- HIGHLIFE#9

PUBLIC DOMAIN PACK AMIGA

Styczeń '91 (nr 1)

- Programy kompresorów danych
- Grafiki Borysa Vallejo
- Prezentacja najlepszych muzyczek
- INTUITRACKER

Luty '91 (nr 2)

- Request player; Multi ripper
- 3-rd day; Phantasmagoria - demo
- Master Seka; Virus Ekspert v1.6
- AMOS-programy; Moduły: Killing game show, Upon Me, Let's swing it.

Marzec '91 (nr 3)

- Najnowszy i najlepszy program muzyczny PROTRACKER V1.0 (pakiet programowy)
- Najlepsze muzyczki: NOW WAIT? - DR.AWESOME
- AMOS - procedury
- DEMO grupy REBELES „TOTAL TRIPLE TROUBLE"

Kwiecień '91 (nr 4)

- RUBBER VECTORS - demo
- KEFTALES - demo
- DISK MASTER V3.0
- Moduły muzyczne:
  - > TECHNOSTYLE 2
  - > GALAXY 2
- GRAFIKA - prezentujemy rysunki
  - > RICK PARKS

Maj '91 (nr 5)

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  - > MIAMI VOICE
  - > ANTI ATARI SONG

Czerwiec '91 (nr 6)

- POWER BOOT - własne menu dysku
- DISK CODING SYSTEM - program do zabezpieczania dysków
- Konwerter IFF - ANSI
- AUER NATION - demo
- Moduły muzyczne
- DOCS - opis gry ELWIRA
- LAMER DEFENCE - do wykrywania i niszczenia wirusów
- REWENG GO OF THE LAMER - grafika w trybie D_HAM

Lipiec '91 (nr 7)

- Sanity - demo
- Amiga - Tanx (1Mb) - gra
- Little Beau (1MB) - gra
- There is A Light/Tonid - modules

Sierpień '91 (nr 8)

- Real 3D - demo nowego programu do raytraceing'u
- Moduł Muzyczny XTC STEREO

Wrzesień '91 (nr 9)

- MODUŁY MUZYCZNE dla programu TFMX:
  - > R - TYPE
  - > THE HOUSE OF TECHNO
- VIRUS EXPERT v181 + 143
- BOOT BLOCK'I
  - > BOOTX v 3.80
  - > IMPLODER v 4.0

Październik '91 (nr 10)

- ANARCHY - „THE INSPIRATION IS NONE"
- DUAL CREW - „NEW DIMENSION"
- SANITY - „ELYSIUM"

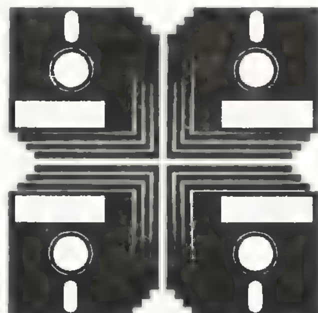
PUBLIC DOMAIN PACK

Listopad '91 (nr 11)

- COMPUTER HEAD - animacja
- MODUŁY POD MEDPLAYER
  -CONFUSED
  -ROCKED
  I WIELE SAMPLI
- SAVE GAME „MONKEY ISLAND"

Grudzień '91 (nr 12)

- GEM X
- BOOTX V. 4.13
- FINAL KIT -monitor
- MEGA-MON
- VARIA



PUBLIC DOMAIN PACK C-64
TAPE NR 1

- TURBO
- SINUSDATA - EDITOR
- FAST CRUNCHER V3
- ANAL S.C. IBEYOND
- VECTOR - VICTORY
- PUZZLENOID+4
- TUNE OF MONTH #1
- NIM
- STRZAŁKA 64+
- LOGO - WRITER V.2.0
- CAN'T TOUCH IKU!
- NTRO PRV
- BONZIEED!!
- ZAX PACKIS
- READ THIS FIRST
- COMMERCIAL BREAK
- 290 SPRITES!
- NOTE - ABOUT
- BAD NEWS NR2
- TO BAD NEWS...
- CONTACT CORNER!
- PROJEKT DUSZKÓW
- SYMPHONY NR14
- SYMPHONY NR15
- SYMPHONY NR16
- SYMPHONY NR17
- SYMPHONY NR18
- SYMPHONY NR19
- CRUISER/GIANTS
- NOTE>ANO<PADUA
- LET'S DYSPI
- FINALTAPE
- MUSIC - SEARCHER

PUBLIC DOMAIN PACK C-64
TAPE NR 2

- TURBO
- PUBL. DOMAIN. INFO
- FONTGRUB 1.0
- DREPTACZ BASIC
- LOAD DIS FIRSY
- MACROASSEMBLER
- TURBOASSEMBLER
- RELOCATOR
- LOGOPAINTER 3!
- REASSEBLER
- SPRITE - EDITOR
- FAST - CRUEL U.2.5
- HIGHLIFE NR5
- AXEL NEWS NR1
- GWIAZDY
- FLIGRAPH 2.2/BML
- NOTE TO FLI V.2.2
- DISKNOTKA/PADUA
- MEGA PACKER/T
- MIST II/ VISION
- TTECHSCR &DYSP
- PLASMA - WORLD
- VECTORBOBS...
- VECTOR - PLOTS
- FLI - UPSCROLL
- BORDER - HIRES
- ROCK AROUND
- FACEWRITER
- CHAR EDIT 2+2
- DISKNOTER
- DESTINATION'91
- CONTACTDEMO/ORE
- FONTEDITOR
- THE END

PUBLIC DOMAIN PACK C-64
TAPE NR 3

- TURBO
- PUBLIC DOMAIN NOTE
- GRAVEYARD NOTES!
- NOTE FROM BEAT!!
- ANONYM SPEAKING!
- SNDK. V3.7/TOPAZ
- AFLI - EDITOR
- NOTER V2.2/TOPAZ
- DLW V1.5/TOPAZ
- CODE - S.MON/PADUA
- OPINION - POLL/PDA
- MOUNTAIN RAID
- PART 1
- PART 2
- PART 3
- PART 4
- PART 5
- FAIRLIGHT 1
- FAIRLIGHT 2
- FAIRLIGHT 3
- FAIRLIGHT 4
- FAIRLIGHT 5
- THE END

PUBLIC DOMAIN PACK C-64
TAPE NR 4

- TURBO
- OUT RUN 2 MUS & SFX
- AFTER BURNER/MON
- FORM.1.SIMULATOR
- 2400 AD.END - TUNE
- NIGHT HUNTER DIGI
- ELEMINATOR MUSIC
- TOMCAT MUSIX/MON
- ZAMZARA TUNE
- DYNAMIX TUNE
- HIGHLIFE NR9
- SNAKES C3
- SNARK C3
- SNERD C3
- WAREHOUSE C3
- STARTREK C3
- TOWER
- SNOOPY
- NEW LAW & ORDER
- FLT/LEGONOTE...
- TERMINAT.2%/FLT
- UNITE!/SYLVIO
- BALL - SCOPE/451
- TRIVIA - GAME MUS.
- RESET - MONITOR
- HOLY SMOKE

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr: 5.09011-400522.7-2511-30-111.0 następujące kwoty: 20.000zł za pojedyńczy zestaw dyskowy dla C-64, 30.000 zł za zestaw programów PD na kasecie, 25.000zł za zestaw dla Amigi.

Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: **imlę i nazwisko, dokładny adres zamawiającego,** skrót **„PDP-64D"** - jeśli zamawiamy zestaw dla C-64 na dyskietce lub **„PDP-64T"** - dla zestawu taśmowego, zestaw dla Amigi prosimy zaznaczać skrótem **„PDP-A"** - dane te prosimy umieszczać na **wszystkich** odcinkach dowodu wpłaty.

W prenumeracie zestawy kosztują: **PDP-64 - 18.000zł** (12 numerów 216 tys. zł), **PDP-A - 22.000 zł** (12 numerów 264 tys. zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Powyższe warunki odnoszą się również do naszych zestawów wydanych w 1991r .

Zestawy taśmowe PDP-64 w 1992r. będą ukazywały się w miarę napływu nowych, ciekawych programów - o czym będziemy informować na łamach naszego pisma.

Zamów nie zwlekaj!

VOICETRACKER V4.0

C-64

Rewelacyjny program muzyczny!



Tylko [illegible] kosztuje fantastyczny edytor muzyczny wykorzystujący ogromne możliwości dźwiękowe komputera Commodore - 64. Oferowany zestaw zawiera dyskietkę lub taśmę magnetofonową [illegible] programem VOICETRACKER V4.0, instrukcję obsługi, oraz - dodatkowo - przykładowe demonstracje muzyczne. **UWAGA! Wersja magnetofonowa tylko 40.000 [illegible]**

Przedsiębiorstwo ABUK posiada wyłączność [illegible] dystrybucję tego programu. Wszelkie kopiowanie programu i powielanie instrukcji jest zabronione. Nabywcy otrzymują rejestrowane kopie programu wraz z prawem nabywania nowych wersji po znacznie obniżonych cenach [illegible] wymiany dyskietki [illegible] razie uszkodzenia. Studiom komputerowym proponujemy zakup hurtowy (przy zakupie powyżej [illegible] kompletów udzielamy [illegible] rabatu).
Chcąc stać się posiadaczem programu VOICETRACKER V4.0 wystarczy dokonać wpłaty 50.000 zł (wersja dyskowa) lub 40.000 zł (taśma) [illegible] konto: Bank PKO SA Bydgoszcz, konto nr: 5.09011-400522.7-136-11-111.0. Na blankiecie prosimy czytelnie podać swoje imię, nazwisko i adres [illegible] dopiskiem „VV4.0" uzupełnionym literką „T" - taśma lub „D" - dyskietka.

W związku [illegible] pojawiającymi się kłopotami w dystrybucji oferowanych przez [illegible] dyskietek i taśm (wynikającymi [illegible] nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet ten może służyć jako zamówienie [illegible] dowód wpłaty dla wszystkich oferowanych przez [illegible] usług: sprzedaż dyskietek [illegible] taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd.

REDAKCJA



Odcinek dla wpłacającego

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla Banku

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla posiadacza rachunku

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla Poczty

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

TREŚĆ ZAMÓWIENIA:

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace. Tematyka programów dowolna. Konkurs rozgrywany jest osobno dla komputerów C-64 i Amiga. Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane). Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu. Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po

500.000 zł każda.

Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma. Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

WSZYSTKICH ZAINTERESOWANYCH NABYCIEM ZALEGŁYCH NUMERÓW

„64 plus 4 & AMIGA"

INFORMUJEMY, ŻE POSIADAMY JESZCZE OGRANICZONĄ ILOŚĆ NUMERÓW OD LISTOPADA 1990R. DO GRUDNIA 1991R.

ZAMÓWIENIA PROSIMY KIEROWAĆ NA ADRES :
Przedsiębiorstwo ABUK sp. z o.o.,
87-200 Wąbrzeźno,
ul. 1 Maja 33.
(Pod tym adresem mieści się dział kolportażu - tam też prosimy przesyłać wszelką korespondencję dotycząca kolportażu czasopisma, dyskietek, taśm itd. Adres redakcji się nie zmienił - patrz stopka.)

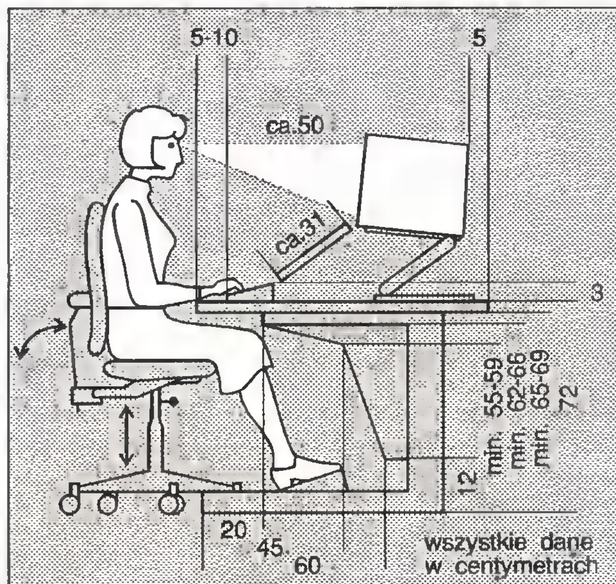
Naukowcy kłócą się czy praca przy komputerze jest szkodliwa czy nie. Jedni wskazują na szkodliwość promieniowania i wydzielających się gazów, inni twierdzą, że wypowiedzi te nie są udowodnione. Jak na to nie patrzeć, zapobieganie nie szkodzi w żadnym wypadku. Własne zdrowie jest tego warte. Kto długo i często siedzi przy komputerze powinien przedsięwziąć odpowiednie środki ostrożności.

DLA WSZYSTKICH

Zdrowie i komputer

Najważniejsze to:

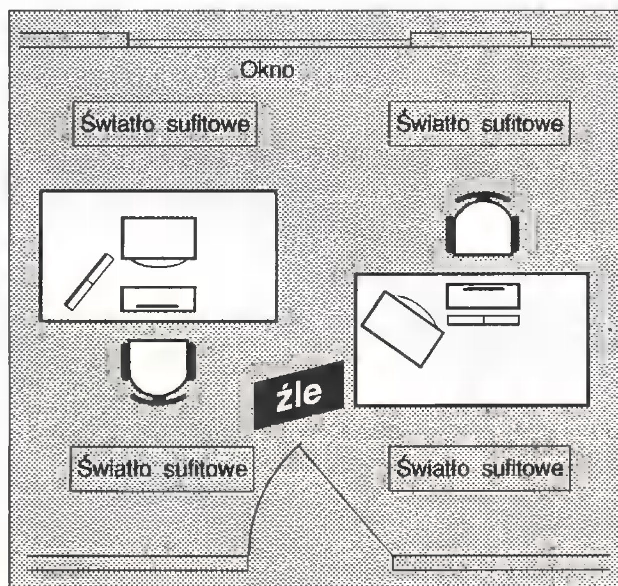
- Monitor powinien znajdować się na wysokości oczu, dzięki czemu nie będziecie zginać karku.
- Nie stawiajcie monitora przed oknem, ani w kierunku okna, ani „pod światło". W pierwszym przypadku będzie zbyt duży kontrast między ekranem a perspektywą, natomiast w drugim przypadku mogą wystąpić odbicia światła od ekranu. Najlepsze miejsce na monitor jest takie, w którym wzrok przebiega równolegle do okna.
- Kontrast między sprzętem a tłem powinien być niewielki.
- Prawidłowy odstęp oczu od monitora wynosi około 50 cm.
- Z uwagi na to, że w wielu monitorach stosuje się specjalne środki ognioochronne, z których - podczas nagrzewania - ulatniają się gazy, ważną sprawą jest wentylacja pomieszczenia. Rzeczą godną pochwały jest to, że coraz więcej producentów monitorów używa innych środków ognioochronnych i przez to są one mniej niebezpieczne.
- Droższe monitory dają lepszy obraz. Jeżeli tylko kieszeń nam na to pozwala, kupujmy dobry monitor.



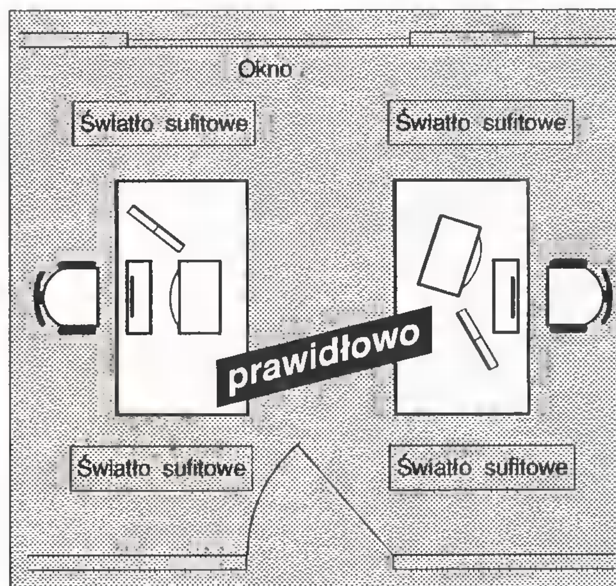
Tak powinno wyglądać ergonomiczne miejsce pracy.

- Przerwy w pracy nie tylko są ważne dla zdrowia, lecz także wpływają na zwiekszenie koncentracji. Róbcie więc często przerwy. Wstańcie i wykonajcie kilka ruchów. Skierujcie wzrok na daleko oddalone przemioty - wasze oczy odprężą sie.

Na podst Amiga-Magazin, nr 9/89.



W tym ustawieniu powstają odbicia światła na ekranie.



Optymalne miejsce dla ustawienia monitora.

AMIGA



Znajdujemy się w Nowym Jorku, jest rok 2004. Od dwóch lat Ziemię opanowali Orbici — okropne i tajemnicze istoty, które przyleciały tu pewnej nocy. Od tego czasu wiele się zmieniło.

Ulice są puste, wiele domów zostało zburzonych lub zamurowanych. Tylko kilka małych sklepów jest jeszcze otwartych. Wszystkie urzędy zostały przejęte przez Orbitów. Ustanowili one następujące prawa:

- zabronione jest opuszczanie swego miejsca zamieszkania,
- zabroniona jest rozmowa z innymi ludźmi,
- zabronione jest pokazywanie swojej twarzy,
- wszyscy zobowiązani są do noszenia brązowych habitów,
- kto nie dostosuje się do tych zakazów i nakazów zostanie zgładzony!

Życie stało sie piekłem.

Wszędzie roznosił się dziwny, przykry zapach. Wszystkim ludziom wszczepiono sondy, za pomocą których Orbici mogą określać ich miejsce pobytu. Ci którzy się bronili, znikneli, prawdopodobnie nie żyją. Aby obronić się przed rebelią Orbici wprowadzili do walki Manhutery (łowcy ludzi). Są to wybrani ludzie wyposażeni we wszystkie techniczne środki, których zadaniem jest zniszczyć lub zatrzymać przeciwników. Do dyspozycji mają również komputery, za pomocą których mogą kontrolować dane każdego człowieka i jego miejsce pobytu (dopóki będzie on na powierzchni Ziemi).

Lecz znalazł się jeden uczestnik ruchu oporu, który odkrył, że sondy Orbitów nie funkcjonują pod Ziemią. Jednemu z nich przydzielone zostały zadania specjalne, na wykonanie każdego ma 24 godziny.

Tak zaczęła się walka w krętych i mrocznych korytarzach kanalizacyjnych.

PRZEBIEG GRY

Idź do szpitala. Tu obejrzyj starannie zwłoki. Na nodze zabitego wisi kartka z jego nazwiskiem, które należy zapamiętać. Podczas przyglądania się twarzom zwłok należy uważać, aby nie zostać zjedzonym przez Orbaby. Wyjdź ze szpitala i idź do kościoła. Zapal tam świecę. Następnie skieruj się do baru Flatbush w północnej części Brooklynu. Tutaj odbywa się gra wideo, która zostanie bez ceregieli przerwana... Rzuć mężczyźnie cztery noże między palce i graj dalej. Przy grze dobrze uważaj. Następnie idź do Prosper Park'u. Tam poszukaj toalety, wejdź do tylniej kabiny po lewej stronie, usiądź i trzykrotnie spuść wodę. W kanalizacji według tego samego schematu jak w grze wideo, szukaj 12 kart, weź je ze sobą i kieruj się do wyjścia. W jaskini weź medalion, następnie udaj sie na Coney Island. Wejdź na środkową budę strzelecką i zestrzel figury w takiej kolejności jak w grze wideo. Wtedy pokaż medalion i weź wygraną kartę danych — przeczytaj ją.

Pierwsze zadanie wykonane. Przy pytaniu o nazwisko podaj cokolwiek. Drugiego dnia odbierz wskazówki i skieruj się na dworzec. Obejrzyj tam wszystko i idź do klubu nocnego Wretched Excess. Tam zwal z nóg czterech punków. Zwróć uwagę na osoby w brązowych habitach, podnieś szybko zagubioną kartę i zabierz ze sobą. Przy Vend—O—Deli notes należy przeczytać w lesie i dalej do Central Parku. Jest tylko jedna droga przez pola minowe. Należy śledzić Target i uważać na łom — będzie później potrzebny. Przy wejściu na pola kierujemy się na prawo i wtedy na karuzelę. Na lewo od karuzeli idziemy przez dwa czerwone krzaki na żółtą drogę na lewo obok studni, dalej na górę ekranu i przy statule na lewo obok drzewa między dwoma krzakami. Teraz przechodzimy między zielonym i czerwonawym drzewem i opuszczamy następny ekran środkiem. Tam idziemy między niebieskim a czerwonym krzakiem i żółtą drogą nad morze. Teraz na lewo od obeliska. między różowym a czerwonym krzakiem.

Oglądamy zwłoki i czytamy nazwisko na chusteczce. Kontrolujemy informacje MAD o Harvey Osborn i udajemy się do jego mieszkania, gdzie odszukujemy klucz i śledzimy ostatni Target do muzeum. Obserwujemy dalej Target i otwieramy drzwi do kanalizacji znalezioną kartą. Ostatnie drzwi otwieramy łomem, potwora uspokajamy medalionem i przechodzimy halę do końca. Przeszukujemy wszystko i notujemy tatuaże zabitych, zabierając ze sobą moduł. To było nasze drugie zadanie.

W trzecim dniu spoglądamy na grabarza i zauważamy nazwisko na nagrobku. Następnie ruszamy do Pawn Shop. Stamtąd bierzemy trzy hafty: gwiazdę, krzyż i kreskę z trzema belkami. Pod ziemią odryglowujemy drzwi kodem (4/1,t1/0/3/1,t2/6/4, t4/2/5) i identyfikujemy zabitych. Ukazuje się człowiek, którego należy pokonać. Po uwieńczonej sukcesem walce bierzemy kartkę i udajemy się do teatru.

Tu oglądamy najpierw plakat po prawej stronie, później idziemy do obrazka, przed którym znajduje się Target. Przesuwamy obrazek i bierzemy kartkę z ukrytego schowka. Przez MAD dowiadujemy się o adres Harry Jonesa i idziemy do jego mieszkania. W radiu znajdziemy Moduł—C. Idziemy do kościoła, zapalamy świece w prawidłowej kolejności i zabieramy moduł A. Wprowadzamy do MAD zaszyfrowane hasło (Phil Cook) i idziemy do Empire State Building.

Włamujemy się do komputera (hasło znajdziemy na kartce z teatru). W okręgu Alpha Security zmieniamy ustawienie ze Special Security na Hall Patrol i w zakresie Betha z Delta Security na Access Security. Następnie wyłączamy komputer i ulatniamy się. Nasze trzecie zadanie wykonane.

Czwarty dzień rozpoczyna się podróżą do szpitala Bellevue. Jak tylko się tam znajdziemy kratę pokonujemy łomem i uciekamy przez labirynt (unikać porażeń prądem!) na wolność. Na dworcu włamujemy się do budynku i wsiadamy do stojącego tam statku kosmicznego. Używamy wszystkich czterech modułów i naciskamy klawisze na pulpicie w następującej kolejności:

— klawisz górny lewy
— klawisz tylni środkowy
— klawisz górny prawy
— klawisz górny środkowy
— dolny prawy
— dolny lewy

Statek kosmiczny startuje. Jeśli wyskoczymy przez otwarte drzwi i przejdziemy przez labirynt to wyjdziemy na wolność. Jeszcze trzeba odpalić bomby w kierunku:

— dworca
— szpitala
— Statuły Wolności
— Empire Statue Building

i gra jest ukończona !

Opr. R.G.

TRACKDISK CZ.1
czyli jak załadować parę bloków

AMIGA

Czasami, gdy piszemy intro, demo czy „jakiś tam" użytek mający obsługiwać dysk po track'ach musimy napisać program obsługi dysku. Na początek wystarczy nam program korzystający z systemowej procedury „trackdisk.device" zajmującej się właśnie obsługą dysku.

Trackdisk.device może wykonywać podstawowe operacje tzn. odczyt, zapis, format, a także wiele innych, mniej lub bardziej przydatnych.

Aby napisać obsługę dysku najpierw musimy utworzyć strukturę IOStdReq, która jest obszarem kontrolnym dla danego urządzenia (console, trackdisk, audio, timer, etc.) i przyporządkować jej odpowiednie urządzenie poprzez jego otwarcie (OpenDevice) dla danej struktury IOStdReq. Poniżej przedstawiony jest krótki program źródłowy składający się z dwóch procedur mających za zadanie inicjację struktury IOStdReq dla trackdisk.device (InitTrackDisk) oraz zamknięcie urządzenia trackdisk (QuitTrackDisk). Pierwszą z nich wywołujemy na samym początku, potem obsługujemy dysk odwołując się tylko do struktury IOStdReq. Procedurę QuitTrackdisk wywołujemy po zakończeniu wszelkich zmagań z dyskiem.

```
Exec:           =   4
FindTask:       =   -294
AddPort:        =   -354
RemPort:        =   -360
OpenDevice:     =   -444
CloseDevice:    =   -450
DoIO:           =   -456

InitTrackDisk:
        move.l  Exec,a6         ; baza exec.library
        sub.l   a1,a1
        jsr     FindTask(a6)    ; odnalezienie zadania,
                                dla którego będzie
                                przyporządkowany port,
                                przez który będzie
                                odbywać się komunikacja
                                z dyskiem
        lea     DiskRep(pc),a5  ; DiskRep jest właśnie
                                tym portem
        move.l  d0,16(a5)       ; adres struktury zadania
                                wpisujemy do struktury
                                portu
        lea     DiskRep(pc),a1
        jsr     AddPort(a6)     ; wykreowanie portu
        lea     DiskIO(pc),a1   ; DiskIO to struktura
                                IOStdReq, która jest
                                najważniejsza
                                w komunikacji z dyskiem
        lea     DiskRep(pc),a5
        move.l  a5,14(a1)       ; struktura IOStdReq
                                musi zawierać adres
                                portu, przez który
                                odbywa się komunikacja
        moveq   #0,d0           ; tutaj podajemy numer
                                urządzenia (w naszym
                                przypadku jest to 0
                                gdyż będziemy
                                obsługiwać stację
                                wewnętrzną Amigi-DF0:)
        moveq   #0,d1           ; znaczniki dla danej
                                struktury (nie ustawiamy
                                żadnych)
        lea     TrackName(pc),a0 ;nazwa urządzenia,
                                które otwieramy
        jsr     OpenDevice(a6)  ; i otwarcie tego
                                urządzenia
        tst.l   d0              ; w przypadku gdy wartość
                                jest różna od zera oznacza
                                to iż urządzenie nie może być
                                otwarte
        bne.s   Error
        rts

QuitTrackDisk:
        move.l  Exec,a6
        lea     DiskIO(pc),a1
        jsr     CloseDevice(a6) ; zamknięcie urządzenia
                                dla naszej struktury
                                IOStdReq
        lea     DiskRep(pc),a1
        jsr     RemPort(a6)     ; usunięcie portu, który
                                wykreowaliśmy podczas
                                inicjacji
        rts
Error:
        moveq   #0,d0
        rts
TrackName:
        dc.b "trackdisk.device",0
        even
DiskIO:
        blk.l   20,0            ; struktura IOStdReq
DiskRep:
        blk.l   8,0             ; struktura Port
```

Jeżeli mamy już zainicjowany dysk możemy rozpocząć zabawę z nim. Zanim jednak opiszę dostępne komendy kilka uwag. Każda z komend wymaga ustawienia pewnych parametrów w strukturze IOStdReq, a więc na początek należy omówić częściowo strukturę IOStdReq:

NAZWA	WIELKOŚĆ	OFFSET	UWAGI
IO_UNIT	(LONG)	24	Zawiera numer urządzenia
IO_COMMAND	(WORD)	28	Zawiera komendę
IO_FLAGS	(BYTE)	30	Znaczniki
IO_ERROR	(BYTE)	31	Numer błędu
IO_LENGTH	(LONG)	36	Długość (bufor, etc.)
IO_DATA	(LONG)	40	Wskaźnik danych
IO_OFFSET	(LONG)	44	Przesunięcie

Teraz aby lepiej zrozumieć zasadę działania IOStdReq mały przykład na wczytanie bloków 880 i 881 pod adres $50000.

```
Read:
        lea     DiskIO(pc),a1
        move.w  #2,28(a1)       ; rozkaz TD_READ
        move.l  #$50000,40(a1)  ; adres bufora
        move.l  #$400,36(a1)    ; długość
                                (2 bloki * $200 bajtów)
        move.l  #880*512,44(a1) ; przesunięcie na dysku
                                do bloku o numerze 880
        move.l  Exec,a6
        jsr     DoIO(a6)
        rts
```

AMIGA

A teraz przedstawię krótki opis kolejnych rozkazów trackdisk.device (w nawiasach podano wartość odpowiadająca temu rozkazowi):

TD_READ (2)
Funkcja umożliwia odczytanie danych zapisanych na dysku.
Wejście:
IO_COMMAND - 2
IO_DATA - wskaźnik bufora, do którego będą wczytywane dane
IO_LENGTH - długość bloku danych do wczytania
IO_OFFSET - przesunięcie na dysku (numer bloku * długość bloku (512 bajtów))
Wyjście:

TD_WRITE (3)
Procedura umożliwia zapis na dysku
Wejście:
IO_COMMAND - 3
IO_DATA - wskaźnik adresu pamięci, od którego rozpoczynamy zapis
IO_LENGTH - długość bloku danych do zapisania na dysku
IO_OFFSET - przesunięcie na dysku (numer bloku * długość bloku (512 bajtów))
Wyjście:

TD_UPDATE (4)
Procedura umożliwia ponowne zapisanie bufora na dysku w celu zabezpieczenia danych. Używamy jej zawsze po zakończeniu zapisu, pozostawiając poprzednie dane w strukturze IOStdReq, a zmieniając tylko numer komendy.
Wejście:
IO_COMMAND - 4
Wyjście:

TD_CLEAR (5)
Wyczyszczenie bufora dysku.
Wejście:
IO_COMMAND - 5
Wyjście:

TD_ADDCHANGEINT (20)
Funkcja dodaje przerwanie do rozszerzalnej listy przerwań, które mogą używać wiele rodzajów driverów.
Wejście:
IO_COMMAND - 20
IO_DATA - zawiera strukturę SoftInt
Wyjście:

TD_CHANGENUM (13)
Funkcja sprawdza stację, w której został zmieniony dysk.
Wejście:
IO_COMMAND - 13
Wyjście:
IO_ACTUAL - zawiera numer napędu, w którym został zmieniony dysk.

TD_CHANGESTATE (14)
Funkcja zwraca aktualny stan danego napędu.
Wejście:
IO_COMMAND - 14
Wyjście:
IO_ACTUAL - różne od zera jeżeli nie ma dysku w napędzie.

TD_FORMAT (11)
Funkcja formatuje dyskietkę w zadanym napędzie dyskowym i wypełnia zformatowane bloki danymi podanymi w buforze. Bufor musi mieć długość przynajmniej jednego bloku. Większy bufor będzie ignorowany. Funkcja nie sprawdza poprawności formatu.
Wejście:
IO_COMMAND - 9
IO_DATA - wskaźnik obszaru zawierającego dane, które będą zapisywane na dysku podczas jego formatowania.
IO_LENGTH - długość obszaru formatowanego, to jest ilość bloków pomnożona przez długość bloku (512 bajtów).
IO_OFFSET - zawiera przesunięcie, od którego będzie formatowany dysk czyli numer bloku mnożymy przez długość bloku.
Wyjście:

TD_GETDRIVETYPE (18)
Ta funkcja zwraca użytkownikowi typ napędu jakiego używa. Wartość zwrócona będzie oznaczała czy dysk jest 3.5 cala, czy 5.25 cala. Czasami jest to potrzebne gdyż napędy 5.25 cala nie mogą wykonywać pewnych operacji.
Wejście:
IO_COMMAND - 18
Wyjście:
IO_ACTUAL - zawiera typ napędu i w zależności od wartości będzie to:
DRIVE3_5 wartość 1
DRIVE5_25 wartość 2

TD_GETNUMTRACKS (19)
Funkcja ta zwraca ilość dostępnych ścieżek dla danego dysku (w przypadku normalnego napędu 3.5 cala wartość ta wynosi 160 ścieżek).
Wejście:
IO_COMMAND - 19
Wyjście:
IO_ACTUAL - zawiera ilość dostępnych ścieżek dla danego napędu dyskowego.

TD_MOTOR (9)
Procedura ta umożliwia użytkownikowi kontrolę nad silnikiem napędu dyskowego, mianowicie na jego włączanie i wyłączanie. Silnik napędu jest włączany zawsze, gdy jest wykonywana jakakolwiek operacja dyskowa.
Wejście:
IO_COMMAND - 9
IO_LENGTH - zawiera informację dotyczącą czy silnik ma być wyłączony czy włączony. Wyłączenie następuje po podaniu wartości 0 natomiast po wpisaniu do tej komórki wartości 1 następuje włączenie silnika napędu dyskowego.
Wyjście:
IO_ACTUAL - zawiera poprzedni stan silnika napędu dyskowego.

TD_PROTSTATUS (15)

Ta funkcja testuje napęd dyskowy, aby stwierdzić czy dysk znajdujący się w tym napędzie jest zabezpieczony przed zapisem czy też nie.

Wejście:
IO_COMMAND - 15
Wyjście:
IO_ACTUAL - zawiera zero, jeżeli dysk nie jest zabezpieczony przed zapisem lub wartość różną od zera, jeżeli dysk jest zabezpieczony. W przypadku gdy w napędzie dyskowym nie będzie dyskietki wtedy do IO_ERROR zostanie wpisana wartość odpowiadająca za błąd TDERR_DiskChanged.

TD_RAWREAD (16)

Funkcja pozwala na wczytanie sektora z dysku w postaci w jakiej się on tam znajduje (czyli tak zwanej RAW). Poszukuje ona wybranego sektora, następnie wczytuje go do bufora zabezpieczonego przez użytkownika. Należy pamiętać o tym, że bufor musi koniecznie znajdować się w pamięci typu chip. Autorzy trackdisk device (również projektanci sprzętu) proponują nie używać tej funkcji, gdyż wprowadzenie ewentualnych modyfikacji może spowodować brak kompatybilności z nowszymi wersjami.

Wejście:
IO_COMMAND - 16
IO_FLAGS - możemy w znacznikach ustawić bit IOTDB_INDEXSYNC, wtedy sterownik rozpocznie odczyt danych z dysku od znacznika synchronizacji (może wystąpić pewne opóźnienie we wczytywaniu danych ze względu na poszukiwanie tego znacznika).
IO_LENGTH - tutaj wpisujemy długość bufora (w bajtach) przeznaczonego na wczytywane dane. Maksymalna możemy ustawić do 32768 bajtów.
IO_DATA - wskaźnik bufora, do którego będą wczytywane dane z dysku.
IO_OFFSET - przesunięcie głowicy - podane w ścieżkach - nie podajemy normalnie (tak jak dla CMD_READ) gdyż sterownik nie wie w jakim formacie dysk jest zapisany.
Wyjście:
IO_ERROR - jeżeli jest różny od zera to oznacza, że wystąpił błąd.

TD_RAWWRITE (17)

Funkcja pozwala na zapis danych na dysku w postaci takiej jak się znajdują w pamięci. Należy jednak uwzględnić, iż nie będzie dokonane żadne przetworzenie danych do zapisu i z tego powodu jest bardzo łatwo popełnić błąd. Dane będą zapisane w formacie MFM.

Wejście:
IO_COMMAND - 17
IO_FLAGS - możemy w znacznikach ustawić bit IOTDB_INDEXSYNC = wtedy sterownik rozpocznie zapis danych na dysku od znacznika synchronizacji. Jednak może nastąpić pewne opóźnienie w zapisie danych ze względu na poszukiwanie tego znacznika.
IO_LENGTH - tutaj wpisujemy długość bufora w bajtach, który chcemy zapisać na dysku. Maksymalna długość jaką możemy ustawić to 32768 bajtów.
IO_DATA - wskaźnik bufora, który chacemy zapisać na dysku.
IO_OFFSET - przesunięcie głowicy - podane w ścieżkach - nie podajemy normalnie (tak jak dla CMD_READ) gdyż sterownik nie wie w jakim formacie dysk jest zapisany.
Wyjście:
IO_ERROR - jeżeli jest różny od zera to oznacza, że wystąpił błąd.

AMIGA

TD_REMCHANGEINT (21)

Funkcja ta usuwa przerwanie dodane za pomocą TD_ADDCHANGEINT.

Wejście:
IO_COMMAND - 21
Wyjście:

TD_REMOVE (12)

Funkcja instaluje i obsługuje przerwanie wywoływane przy wyjmowaniu dysku.

Wejście:
IO_COMMAND - 12
Wyjście:

TD_SEEK (10)

Funkcja ta pozwala na ustawienie głowicy w danym miejscu dysku. Gdy są wywoływane dane operacje obsługi dysku to procedura ta jest wywoływana automatyczne, jednakże jeżeli znamy pozycję skąd będzie wykonywana następna operacja możemy przesunąć tam głowicę wcześniej, oszczędzając trochę czasu.

Wejście:
IO_COMMAND - 10
IO_OFFSET - tutaj wpisujemy pozycję, na którą chcemy przesunąć głowicę.
Wyjście:

To już wszystkie procedury dostępne w trackdisk.device.

Zachęcam wszystkich do eksperymentowania z nimi i częstszego używania niż własnych loaderów, gdyż zapewnia to większą kompatybilność.

Marcin "Duddie" Dudar

AMIGA

KĄCIK POCZĄTKUJĄCEGO KODERA cz.11

Dzisiejszy kącik poświęcimy opisaniu poszczególnych assemblerów, ich zalet oraz wad. Będą to najpopularniejsze z nich: MasterSeka 1.80, AsmOne 1.05, ArgAsm 1.0, Devpac 3.02. Asemblery będą oceniane w skali dziesięcio-punktowej według następujących kryteriów:

- **edytor czyli wygoda edycji,**
- **szybkość edytora,**
- **asemblacja czyli możliwości asemblera,**
- **szybkość asemblacji,**
- **możliwości czyli pozostałe rzeczy, które może wykonywać asembler oraz**
- **ocena ogólna.**

■ MasterSeka 1.80

Gdy powstały komputery Amiga i Atari ST firma Kuma wypuściła dla tych komputerów bardzo prosty asembler i była nią właśnie Seka. Może ze względu na prostotę obsługi a może z innych przyczyn zainteresowali się nią koderzy i postanawiali udoskonalać. Od czasu Seki 1.5 (ostatnia wersja wypuszczona przez firmę Kuma) powstało wiele wersji coraz wygodniejszych w obsłudze.

W 1988 roku gruntownych przeróbek dokonał ACE z grupy Megaforce i powstała wtedy Seka 2.0. Następnym krokiem do przodu była Seka 3.2 zrobiona przez Promaxa (znanego jako Rune Gram-Madsen). Jednak największą popularność spośród asemblerów Seka-podobnych zdobyła sobie MasterSeka. Początkowo kodowana przez Buddhę z grupy Spreadpoint. Ostatnia wersja nosząca numer 1.80 została zrobiona przez Corsaira z grupy SkidRow.

MasterSeka jest zdecydowanie najbardziej wygodnym z asemblerów chociaż ma wiele wad. Największą jest to, że nie można utworzyć więcej niż dwie sekcje czyli hunki, a na dodatek sam proces asemblacji przebiega bardzo wolno. Edytor MasterSeki jest najwygodniejszym ze wszystkich. Wszelkie funkcje są dostępne poprzez wciśnięcie jakiegokolwiek z klawiszy Amiga z zadanym klawiszem lub przez kombinację z klawiszem kontrol, bądź też poprzez wybranie opcji z menu. Jego wadą jest - niestety - to, że przy otworzeniu obszaru roboczego powyżej dwudziestu trzech linii zaczyna „głupieć" gdy wykonujemy przesunięcie tekstu o całą stronę.

Ocena:

Edytor:	10
Szybkość edytora:	5
Asemblacja:	7
Szybkość asemblacji:	3
Możliwości:	7
Ogólnie:	6

■ AsmOne 1.05

Asembler - autorstwa Rune Gram-Madsena czyli inaczej Promaxa - jest nieco zbliżony do Seki. Największą z zalet AsmOne jest zainstalowanie w nim bardzo dobrego debugger'a i całkiem dobrego (choć niezbyt rewelacyjnego) monitora. Edytor mimo, że jest dobry i szybki nie należy do wygodnych. Brakuje w nim wielu funkcji, do których przyzwyczajony jest użytkownik Seki bądź CED'a. Asemblacja jest bardzo szybka jednak nie pozbawiona błędów i - na przykład - możemy asemblować tekst, w którym etykiety kończą się dwukropkiem lub bez dwukropka, ale nie możemy mieć ich jednocześnie. Cuda zaczynają się dziać gdy popełnimy błąd, którego AsmOne nie wykrywa, na przykład gdy odnosimy się względem licznika rozkazów do danych znajdujących się w innej sekcji. Asemblacja przebiegnie bezbłędnie natomiast przy nagrywaniu pliku możemy uzyskać komunikat 'No object written' (jest to najlepszy przypadek najczęściej jednak Amiga wykonuje małe Guru).

Ocena:

Edytor:	6
Szybkość edytora:	10
Asemblacja:	5
Szybkość asemblacji:	8
Możliwości:	9
Ogólnie:	8

■ ArgAsm 1.0

Program ten, wydany przez firmę Argonaut software, jest najszybszym z asemblerów dostępnych na Amigę, jednak jeżeli chodzi o edytor to... lepiej się nie wypowiadać. Jest on wolny a na dodatek nie posiada żadnych możliwości. Za pomocą tego programu nie można nic napisać. Asemblacja - jak już wspomniałem - jest bardzo szybka i przy długich programach trwa kilka sekund, natomiast jeżeli chodzi o możliwości procedury asemblującej to są one tragiczne. Gdy popełnimy jakiś błąd to bardzo rzadko zdarza się uzyskać komunikat o błędnej instrukcji - najczęściej jedynym komunikatem jest migająca dioda.

ArgAsm nie posiada żadnego debuggera ani monitora i może asemblować tylko na dysk co bardzo obniża jego użyteczność.

Twórcy tego asemblera starali się z niego zrobić pewną alternatywę dla Devpaca, ale nie udało im się to, poza tym jeżeli tworzyć jakąś alternatywę to dla asemblera lepszego (jak na przykład AsmOne jest alternatywą dla Seki).

Ocena:

Edytor:	2
Szybkość edytora:	2
Asemblacja:	2
Szybkość asemblacji:	10
Możliwości:	1
Ogólnie:	3

■ Devpac 3.02

Firma HisSoft chyba jako pierwsza wypuściła na rynek asembler dla Amigi - było to w roku 1985 - był to Devpac (znany użytkownikom mniejszych maszyn). Wersje Devpac'a od 1.0 do 2.1 były - ogólnie rzecz biorąc - niewypałami, dopiero wersja 3.02 wypuszczona na rynek w tym roku pokazuje, że programiści z HiSoftu nie są wcale tacy źli. Edytor zmienił się niewiele, ale teraz można go już używać

(chociaż nie należy do genialnych), natomiast asemblacja działa i - trzeba dodać, że jest to najlepsza asemblacja w asemblerach dla Amigi (chociaż nie jest rewelacyjnie szybka).

Możemy dokonywać wielu operacji: asemblacji dla różnych procesorów, koprocesorów, a co najważniejsze asembler ten ma wbudowane bardzo dobre procedury optymalizacji oraz bardzo dobrą wykrywalność błędów. Jako debugger został użyty MonAm 3.02 i trzeba przyznać, że jest to dobry debugger o dużych możliwościach.

Ocena:

Edytor:	6
Szybkość edytora:	4
Asemblacja:	10
Szybkość asemblacji:	8
Możliwości:	7
Ogólnie:	7

Asemblerów wydanych zostało znacznie więcej, ale tylko te nadają się - moim zdaniem - do jako takiej pracy.

Marcin "Duddie" Dudar

PUBLIC DOMAIN PACK

nr 14 (LUTY'92)

Włączamy komputer, wkładamy do napędu dyskietkę i po chwili ukazuje się nam INTERFERON PRO!

Po uruchomieniu mamy do dyspozycji następujące opcje:

- F1 - FAST MEM OFF - wyłączenie dodatkowej pamięci typu fast (o ile taką posiadamy).
- F2 - FAST MEM ON - włączenie pamięci fast (o ile wcześniej ją odłączyliśmy programowo).
- F3 - TOGGLE DRIVES - odłączanie i włączanie dodatkowych stacji dysków (oczywiście trzeba je posiadać).
- F4 - HARD RESET - całkowite wyczyszczenie pamięci.
- F5 - 60HZ OFF - przełączenie pracy komputera z systemu NTSC na PAL (często objawia się to nieco uciętym z dołu ekranem).
- F6 - 60HZ ON - włączenie systemu NTSC, ale „prawidłowo" tzn. ekran zostaje troszkę rozciągnięty, a przerwania komputera wykonywane nieco szybciej (większość gier zrobiona jest w tym systemie i po włączeniu będą wyglądały prawidłowo).
- F7 - TOGGLE LED - włączenie i wyłączenie power'a (w nowych amigach przygaszenie).
- F8 - MOVE 1/2 CHIP - wyjście z programu z ustawionym 1/2 mega pamięci CHIP.
- F9 - INSTAL DF0: - nagranie bootblock'u na dysk, aktualnie włożony do napędu.
- F10 - DISC COPY - kopiowanie dyskietek (taki mały X-COPY). Jeżeli program zgłosi się kolorowymi paskami znaczy to, że w pamięci w danym momencie znajduje się wirus. Lewym przyciskiem myszy usuwamy go, natomiast prawym zostawiamy w spokoju.

AMIGA

Po przejściu przez bootblock komputer kontynuuje wczytywanie zawartości dyskietki i po krótkiej chwili ukazuje się spis programów, które znajdują się na PUBLIC DOMAIN PACK'u. Jak wynika ze spisu, wyboru dokonujemy używając klawiszy funkcyjnych.

- F1 - do pamięci wgrywamy najnowsze demko, dobrze znanej z wydawanego magazynu dyskowego pod tytułem ZINE, grupy BRAINSTORM. Demko BLITTER MIRACLE (ba taki nosi tytuł) stoi na najwyższym światowym poziomie. Autorzy umieścili w nim wszystkie najpopularniejsze w ostatnim czasie efekty, na przykład zdjęcia obracane wektorowo, bardzo szybki fraktal. Są to tylko niektóre z atrakcji. Program szokuje nas również bardzo ładną muzyczką CHESTER'a.
- F2 - LITTLE VECTOR PREVIEV - grupy COMPLEX. Wydawałoby się, że wektorówka już dawno zaczęła nas nudzić, ale to demo już w pierwszej chwili „zwala z nóg". Tak szybkiej wektorówki jeszcze nie było !!! Obiekty rodem z gwiezdnych wojen poruszają się płynnie i szybko. Warto obejrzeć!
- F3 - ACTION DIRECT FLYING DOTS INTRO - intro polskiej grupy. Podstawowy efekt oparty jest na falujących napisach. Efekt ten może być wyzwaniem dla innych koderów, gdyż poruszających się punktów jest aż 2100. Miłą niespodziankę będą mieli również posiadacze ACTIONREPLAY'a. Próby spowolnienia działania tego intra nie spowodują zmian w szybkości przesuwania się scroll'a umieszczonego w dolnej części ekranu.
- F4 - FRIENDSHIP - to moduł, który możemy uruchomić również pod PRO TRACKER'em. Został napisany podczas COPY PARTY w Gdyni przez prawie wszystkich najlepszych polskich muzyków. Są w nim kawałki napisane przez MR.ROOT'a, RAF'a, XTD, PETERS'a, KADI'ego i MR.PAVEL'a. To jest chyba pierwsza w polsce muzyka napisana przy takiej współpracy.
- F5 - NUKE SADDAM 1.4 - SADDAM to chyba ostatnio najwredniejszy z wirusów. VIRUS ten uaktywnia się automatycznie po włożeniu zainfekowanej dyskietki. NUKE SADDAM jest jednym z nielicznych programów, które potrafią z nim się uporać. Ogromną zaletą programu jest fakt, że NUKE naprawia dyskietki na których SADDAM zdążył wyrządzić już jakieś szkody. Oprócz SADDAMA program wykrywa i kasuje niemal wszystkie wirusy, które "urzędują" w „diskvalidatorze".
- F6 - THIEF RIPPER 2.0 - kolejny program wyszukujący w pamięci moduły muzyczne. Ten chyba jednak - do czasu ukazania się MULTI RIPPER'a 3.0 - będzie najlepszy. Wyszukuje bowiem muzyczki napisane w prawie wszystkich programach muzycznych. Nawet moduły z NOISE PACK'owane czyli przekonwertowane z formatu programu muzycznego NOISE TRACKER'a na nowy, w którym muzyczki są mocno spakowane, a player routine odtwarzający je zajmuje znacznie mniej czasu mik-



AMIGA

roprocesora. Wyszukiwanie w pamięci modułów NOISE PACKER'a jest tym bardziej przydatne, iż wszystkie najlepsze grupy na świecie, takie jak PHENOMENA, ANARCHY, RAZOR 1911 od jakiegoś czasu robią ze swoimi muzyczkami tą operację.
A oto lista innych programów muzycznych, których moduły wyszykuje w pamięci THIEF: SID MON I SID MON II, FUTURE COMPOSER 1.3, FUTURE COMPOSER 1.4, JAM CRACKER, DELTAMUSIC, MARK II, SOUDNSYSTEM DAWID, WHITTAKER MUSIC, NOISE TRACKER, PRO TRACKER, NOISE PACKER.

- F7 - DISK MASTER 3.05 - Najlepszy z diskmasterów! Jest to wersja poprawiona przez Maćka (KATHARSIS), który usunął z niej sporo błędów (jego wersja 3.2 miała jedynie zmienioną grafikę, a błędy pozostawione z 3.0). Program jest dosyć prosty w obsłudze dlatego też nie zajmę się jego opisem (może przy innej okazji). Dzięki poprawieniu szeregu opcji program ten warty jest używania i polecam go wszystkim (nie posiadającym twardego dysku) bardziej niż np. DISK MASTERA 2.0 czy OPUSA 3.2. Program ma sporo użytecznych opcji: zmiana systemu wizji z PAL na NTSC (i odwrotnie), odgrywanie modułów NOISE TRACKERA (także spakowanych POWER PACKEREM). Rozpakowuje również pliki tekstowe jeśli chcemy je przeczytać. Ma możliwość kopiowania całych dyskietek i ich sprawdzania. Posiada wbudowany VIRUS KILLER. DISK MASTEREM możemy również przeglądać dyski „po trackach" - opcja BLOCK EDITOR. Po za tym program ma niepodważalną zaletę w stosunku do „konkurentów" - jest bardzo krótki i wczytuje się bardzo szybko (czego nie można powiedzieć np. o OPUSIE).



- F8 - ZIG ZAG#3 - obrazek tytułowy. Zapewne wielu z Was pamięta KEBAB'a. Był on pierwszym magazynem dyskowym wydawanym po polsku. Nie był jednak ostatni! W tej chwili wydawane są cztery nowe magazyny tj. ALA MA KOTA, FAT AGNUS, NEXT LIFE i oczywiście ZIG ZAG.



NINJA /ATD

Przypominamy zawartość zestawu PDP Amiga nr 13 (styczeń '92):

- Super Duper 2.01
- Sanity Copy
- Noise Packer 3.00
- Ham Sharp
- Mostra

oraz, jak zwykle, graficzne i muzyczne

- dema.

O zasadach nabywania dysków PD piszemy na str. 16.

W naszej „ofercie gwiazdkowej" zapowiadaliśmu, że wśród wszystkich, którzy zamówią komplet dysków PDP za rok 1991 na Amigę

rozlosujemy 10 programów D-Mon Professional.

Oto lista osób do których „uśmiechnęło się" szczęście:

- Maciej Sampara, Poznań
- Dariusz Przeszłowski, Gdańsk
- Tomasz Skiba, Ruda Śiąska
- Bogdan Begier, Kołobrzeg
- Tomasz Śliwa, Kowary
- Marian Kopala, Gliwice
- Zenon Grodzki, Łódź
- Bartłomiej Deptuła, Szczecin
- Andrzej Kruszewski, Kalisz Pomorski
- Jacek Konopko, Białystok.

GRATULUJEMY!

ARP LIBRARY CZ.5

AMIGA

Dzisiejszy odcinek opowieści o arp.library jest już ostatnim z tego cyklu. Mam nadzieję, że cykl ten pomógł zainteresowanym dokładniej poznać tą bibliotekę.

GADS - **Wykonuje rozbiór standardowej linii z komendami**

Ilość = GADS ("Linia", długość, "help", arg, "wzór")
D0 A0 D0 A1 A2 A3

Ta funkcja zapewnia rozbiór standardowej linii z komendami AmigaDOS. używając jako wzorca AmigaDOS. Procedura zapewnia przyjęcie wszystkich znaków poprzedzonych kodem "escape".

Wejście:

Linia - wskaźnik linii taki sam jak otrzymaliśmy z AmigaDOS przy wejściu do programu.

Długość - długość linii taka sama jak przekazana przez AmigaDOS.

Arg - wskaźnik tablicy długich słów, w której składowane będą rezulatаty rozbioru linii. Musi zawierać tak dużo słów jak dużo argumentów wpiszemy do wzorca. Jednak musi zawierać przynajmniej jeden element.

Wzór - wskaźnik do wzoru utworzonego w stylu AmigaDOS. Funkcja przyjmuje, że wzór jest podany bezbłędnie i jest właściwie ukształtowany to znaczy nie ma w nim spacji, tabulatorów i podobnych rzeczy oraz, że jest zakończony zerem. Zabrania się kończenia wzoru znakiem ':'. Wzór może zawierać zarówno małe jak i duże litery.

Help - opcjonalny wskaźnik tekstu wyświetlany gdy użytkownik wpisze znak zapytania. Tekst może być dowolnej długości i może zawierać dowolne znaki, ale musi być zakończony zerem. Wskaźnik ten może być równy zero jeżeli nie chcemy dostarczyć żadnej pomocy.

Wyjście:

Ilość - rejestr zawiera ilość wprowadzonych argumentów lub -1 jeżeli wystąpił błąd w linii komendy. Funkcja automatycznie sprawdza składnie linii.

Jeżeli "ilość" jest różna od zera to tablica "arg" zawiera rezultaty rozbioru linii. Każdy argument (jeżeli jest obecny) będzie reprezentowany na pozycji w tablicy argumentów zarezerwowanej we wzorze. Na przykład: jeżeli we wzorze wystąpi "DIR,TO" to DIR zawsze będzie obecny na pierwszej pozycji, a TO na drugiej, bez względu na to jak użytkownik je dostarczył.

Jeżeli "ilość" jest równa zero to tablica "arg" będzie zawierała wartości z jakimi została zainicjowana.

Jeżeli "ilość" jest mniejsza od zera to oznacza, że wystąpił błąd na pierwszej pozycji w tej tablicy znajduje się wskaźnik do tekstu opisującego ten błąd. Program może wydrukować ten ciąg a następnie zakończyć działanie.

Błąd może wystąpić także wtedy gdy użytkownik pominął wymagany argument, ale gdy długość linii komendy jest niezerowa.

SyncRun - **Uruchamia zadanie i oczekuje na jego zakończenie**

Wynik = SyncRun ("Nazwa", "Argumenty", Input, Output)
D0 A0 A1 D0 D1

Procedura SyncRun ładuje program z dysku przeszukując ścieżki ustawione przez komendę Path, uruchamia go i powraca z wartością zwracaną przez program po tym jak został on wykonany. Można ustawić standardowe wejścia i wyjścia dla nowego procesu ale jeżeli zostanie podane zero to zadane przejmie wejścia i wyjścia z zadania z którego zostało uruchomione. Ta funkcja jest utworzona, aby zapewnić jak największe bezpieczeństwo uruchamiania zadanych programów. Niektóre z programów (te napisane w BCPL'u) mają kłopoty w przejmowaniu argumentów gdy są uruchamiane z poziomu tej funkcji. Wszelkie inne programy działają poprawnie gdyż nie posiadają błędu zawartego w programach pisanych w BCPL'u.

Wejście:

Nazwa - nazwa programu, który chcemy wczytać i uruchomić.

Arg - argumenty, które chcemy przekazać do uruchamianego programu jednak nie następuje rozbiór tych danych i dlatego nie powinniśmy się spodziewać efektów z przekazywania wyników za pomocą io_redirection.

Input - standardowe wejście dla nowego zadania. Jeżeli zero to zostanie pobrane z aktualnego zadania. Wartość musi być osiągnięta i dostarczona przez programistę.

Output - standardowe wyjście dla zadania gdzie będą umieszczane wszelkie wyniki z tego zadania. Jeżeli zero to zostanie pobrane z aktualnego zadania. Wartość musi być osiągnięta i dostarczona przez programistę.

Wyjście:

Wynik - jeżeli wynik jest dodatni bądź zerowy to jest to wynik z wykonanego zadania. Zero oznacza pełny sukces. Natomiast gdy osiągnięta wartość jest negatywna oznacza to iż funkcja nie może wykonać bądź załadować programu.

TackOn - **Poprawne dodanie nazwy zbioru do nazwy ścieżki.**

TackOn("NazwaŚcieżki", "Nazwa Zbioru")
A0 A1

Ta funkcja poprawnie dodaje nazwę zbioru do istniejącej nazwy ścieżki wprowadzając właściwy separator (znak dwukropka bądź dzielnika). Użytkownik jest odpowiedzialny za zapewnienie wystarczająco dużego bufora. Funkcja nie sprawdza czy pole zarezerwowane na nazwę ścieżki uległo przepełnieniu.

Wejście:

NazwaŚcieżki - wskaźnik nazwy ścieżki, która będzie uzupełniona nazwą zbioru.

NazwaZbioru - wskaźnik nazwy zbioru, którą będzie uzupełniana nazwa ścieżki.

I to już wszystkie procedury zawarte w arp.library za wyjątkiem jednej, a mianowicie FileRequest, która została omówiona we wrześniowym numerze 64 PLUS 4 z ubiegłego roku, w artykule "FILE REQUESTERS".

Wszystkim, którzy piszą swoje programy polecam stosowanie arp.library jako bardzo dobrej alternatywy dla dos.library.

Kończąc tę serię zachęcam do czytania kolejnych artykułów poświęconych bibliotekom.

Marcin "Duddie" Dudar

AMIGA

Program Power Packer - autorstwa Nico Francois - stał się pewnym standardem jeżeli chodzi o pakowanie danych.

Pod koniec roku 1991 autor wydał wersję 4.0 tego programu i trzeba przyznać, że jest ona bardzo udana i zawiera wiele dobrych elementów.

POWER PACKER PROFESSIONAL V4.0

W najnowszej wersji PowerPacker'a wykorzystane są dwie biblioteki tego samego autora o nazwach 'reqtools.library' oraz 'powerpacker.library'. Wykorzystując te dwie biblioteki możemy stworzyć własne programy bardziej eleganckie oraz uczynić je bardziej kompatybilnymi z innymi.

Jeżeli - na przykład - algorytm kompresji danych za pomocą PowerPackera zostanie unowocześniony to nie będziemy musieli wprowadzać zmian do programu, wystarczy tylko przegrać nową bibliotekę.

Biblioteka 'powerpacker.library' służy do dekompresji i kompresji danych. Można za jej pomocą wczytywać skompresowane dane do naszych programów a także nasze programy mogą nagrywać skompresowane dane.

Biblioteka 'reqtools.library' zapewnia nam bardzo eleganckie requestery w tym także bardzo dobry File Reqeuester. Samo wykorzystanie tej biblioteki może w pewnym momencie przerazić użytkownika, gdyż autor przekazywanie parametrów dla requesterów oparł na zasadzie tag'ów czyli pewnych list rozkazów, ale mogę zapewnić, iż jest to bardzo wygodna a zarazem bardzo dobra metoda.

Nowością PowerPacker'a jest także zainstalowanie w nim handlera dla ARexx'a, który pozwala na sterowanie PowerPacker'em w każdym momencie. Bardzo bogaty zestaw komend zapewnia dostęp do wszystkich funkcji, do których mamy dostęp normalnie. Dla wersji Kickstarta 2.0 ARexx ma o wiele bogatsze możliwości, jak na przykład instalacja 10 komend na stałe oraz modyfikacja ich pod PowerPackerem.

Należy przyznać, iż PowerPacker jest programem bardzo dobrym i stał się prawdziwym standardem.

Marcin "Duddie" Dudar

KURS JĘZYKA C cz. 6

W poprzednim odcinku mówiliśmy o strukturach. Pozostały nam jeszcze do omówienia słowa kluczowe języka C.

Zapewne wielu z Czytelników zastanawia się, dlaczego tak istotną rzecz pozostawiłem na koniec. Wynikało to z prostego faktu. Przedstawienie dużej ilości rozkazów języka mogłoby wywołać klasyczny mętlik w głowie i zaciemnić niektóre osobliwości języka. Jak się okazało, można poznać praktycznie CAŁY język C, nie znając wszystkich słów kluczowych. Warto bowiem w tym miejscu podkreślić jedną sprawę. Dobrym programistą może zostać ta osoba, która zrozumiała zależności przedstawione w poprzednich częściach cyklu.

Nadrabiamy więc „zaległości" w znajomości słów kluczowych C.

Oto one wszystkie w kolejności alfabetycznej:

auto	double	long	switch
break	else	register	typedef
case	extern	return	union
char	float	short	unsigned
continue	for	sizeof	while
default	goto	static	
do	int	struct	

Większość z nich została przedstawiona w poprzednich częściach. Omówienia wymagają jeszcze: **break, case, continue, default, goto, sizeof, switch, typedef i union**.

Zacznijmy od tego, co zawsze wywołuje największe kontrowersje: w C istnieje instrukcja goto! Jej użycie następuje według schematu:

```
    goto etykleta;
    .
    .
    .
etykieta:
```

gdzie etykietą jest dowolna nazwa (oczywiście nie pokrywająca sie z nazwą jakiejś zmiennej czy funkcji). Etykieta kończy sie znakiem ':' (dwukropek). Program po napotkaniu instrukcji goto realizuje skok do tego miejsca programu, w którym umieszczono etykietę.

Fachowcy od BASIC'a niech lepiej zapomną, że coś takiego istnieje. W programie, którego tekst zajmuje 100 KB instrukcja **goto** powinna sie znaleźć nie więcej niż... zero razy. **Goto** należy unikać tak, jak się tylko da, choć są

przypadki, gdy bez niej bardzo trudno się obyć. Klasyczny przykład to wychodzenie z wnętrza zagnieżdżonych pętli itp.

Instrukcje **break** i **continue** służą do obsługi pętli.

Instrukcja **break** powoduje przerwanie wykonywania aktualnej pętli. Zwrócić należy uwagę na fakt, że instrukcja ta przerywa działanie tylko jednej pętli - tej, w której się bezpośrednio znajduje. Oto przykład:

```
main()
{ int x, y;
   for ( x=0; x; x++ )
   for ( y=0; y; y++ )
  {    if ( y 4 )
   break;
   printf( "\nX = %d, Y = %d", x, y );
   }
}
```

Wykonywane są dwie pętle: x oraz y. Gdy wartości y staną się większe od 4, a więc przy y równym 5, następuje zakończenie wykonywania pętli y. Ponieważ jednak znajdowała się ona wewnątrz pętli x, nastąpi ponowne wykonanie pętli y.

Z kolei instrukcja **continue** powoduje zaniechanie wykonywania dalszej części pętli i powrót do jej początku.

Poniższy program przedstawia sytuację, w której liczba 5 jest dzielona przez kolejne liczby całkowite od -10 do 10. Ponieważ dzielenie przez zero nie prowadzi do niczego dobrego, argument równy zeru zostaje pominięty:

```
main()
{ int x; float y;
   for ( x=-10; x; x++ )
  {  if ( x == 0 )
      continue;
     y = 5.0 / (float) x;
   } printf( "\nI nie ma byka!\n" );
}
```

Ze względu na fakt korzystania z procedur zmiennoprzecinkowych, powyższy przykład należy linkować także z biblioteką m.lib:

```
cc prog    ln prog -lc -lm.
```

Instrukcje **switch, case, default i break** są ze sobą ściśle związane. Służą one do obsługi instrukcji **switch**, która ma następujący format:

```
      switch ( zmienna )
{        case wartosc1:
          instrukcje1;
        case wartosc2:
          instrukcje2;
                        .       .
             .
        deafult:
          instrukcje3;
}
```

Zasada jest prosta. Jeśli „zmienna" przyjmie wartość równą „wartość1", wtedy wykonaj „instrukcje1", jeśli jest to „wartość2", wtedy wykonaj „instrukcje2" itd. Słowo **case** obowiązkowa poprzedza wartość zmiennej.

Uwaga! Wyrażenie po **case** musi być stała! Słowo **default** jest nieobligatoryjne i oznacza: „w pozostałych przypadkach wykonaj...". Istotną rzeczą jest zrozumienie działania rozkazu **break** w tym przypadku. Otóż jeżeli na końcu ciągu instrukcji po **case** nie napiszemy **break**, program zacznie wykonywać instrukcje należące do następnego **case** (lub **default**) - o ile takie instrukcje istnieją. Poniższy przykład powinien rozwiać wątpliwości:

```
main()
{ int x;
 for ( x=0; x; x++ )
   switch ( x )
   {      case 0:
      printf( "\nTo zero!" );
      break;
    case 1:
      printf( "\nTo jeden!" );
      break;
    case 2:
      printf( "\nA kuku!" );
/* Ponieważ nie ma break,
program wykona dla x = 2
wykona także poniższą instrukcję:*/
    case 3:
      printf( "\nTo wartosc: %d", x );
      break;
    default:
      printf( "\nKilka ciekawych danych: %d", x );
   }
}
```

AMIGA

Na marginesie mogę dodać, że „zjedzona" instrukcja **break** po **case** jest częstokroć przyczyną nie wytłumaczalnego zachowywania się programu .

Często zachodzi konieczność uzyskania informacji o wielkosci określonej danej. Do tego celu używa się operatora **sizeof()**. Operator ten ma format:

```
sizeof( zmienna )
```

i jako wynik zwraca ilość bajtów, które zajmuje podana zmienna. O ile bowiem nietrudno dowiedzieć się, jaki jest rozmiar zmiennej typu int, to trzeba się chwilę zastanowić, ile bajtów zajmuje jakaś rozbudowana struktura. Oto przykład:

```
main()
{ int x;
  printf("\nRozmiar zmiennej int w bajtach: %d.",
sizeof( x ) );
}
```

Ciekawą rzeczą są unie. Unie definiuje się analogicznie jak struktury, lecz zamiast słowa struct używa się union. Np.

```
union test
{ int x; douhle y; };
union test Test;
```

Zdefiniowaliśmy unie typu test, a następnie zadeklarowaliśmy zmienną Test będącą unią typu test. Otóż unia jest zmienną, która może być traktowana jako zmienna różnego typu, w zależności od definicji. W powyższym przykładzie zmienna Test może być typu int lub też typu double. Jest to po prostu sposób na tworzenie „uniwersalnych" zmiennych. Unia ma zawsze rozmiar największego elementu, jakim może być. Stosowanie unii należy do rzadkich przypadków, jeśli się je jednak stosuje, wtedy należy pamiętać o tym, że tylko programista wie, jaki typ zmiennej jest zawarty aktualnie w unii. Używanie unii wymaga więc odrobiny doświadczenia i dużej uwagi.

Słowo kluczowe typedef jest używane do tworzenia nowych nazw typów zmiennych. Jak łatwo zauważyć, programy na Amigę wykorzystują różne "dziwne" typy danych. Zostały one zdefiniowane przy pomocy instrukcji typedef w celu łatwiejszego rozróżnienia, do czego określone dane służą. Oto definicje typów (zawarte w pliku exec/types.h):

```
typedef void                  *APTR;
typedef long                  LONG;
typedef unsigned long         ULONG;
typedef unsigned long         LONGBITS;
typedef short                 WORD;
```

```
typedef unsigned short      UWORD;
typedef unsigned short      WORDBITS;
typedef signed char         BYTE;
typedef unsigned char       UBYTE;
typedef unsigned char       BYTEBITS;
typedef short               RPTR;
typedef unsigned char       *STRPTR;
typedef short               SHORT;
typedef unsigned short      USHORT;
typedef short               COUNT;
typedef unsigned short      UCOUNT;
typedef ULONG               CPTR;
typedef float               FLOAT;
typedef double              DOUBLE;
typedef short               BOOL;
typedef unsigned char       TEXT;
```

Sens jest prosty. Do czego służy zmienna

unsigned char *test;

domyślamy się, lecz nie jesteśmy pewni przed analizą programu. Może to być wskaźnik na tekst, przydzielana pamięć czy mapę bitów. Natomiast

STRPTR test;

oprócz zwięzłości od razu precyzuje nam, że chodzi o wskaźnik na tekst.

Oprócz słów kluczowych, operatorów, struktur i tym podobnych, w sumie standardowych cech, C wyróżnia fakt istnienia preprocesora. Rozkazy (dyrektywy) preprocesora umieszcza się po znaku '#' i mogą to być:

```
#include nazwa_pliku
#include "nazwa_pliku"
#define nazwa ciag_znakow
#undef nazwa
```

oraz grupa dyrektyw kompilacji warunkowej:

```
#if      wyrazenie
#ifdef   wyrazenie
#ifndef  wyrazenie
#else
#endif
```

Rozkaz #include nazwa_pliku oznacza: dołącz do kompilowanego programu plik o nazwie nazwa_pliku, który znajduje się w standardowym katalogu zawierającym pliki nagłówkowe. Np.

#include /types.h

Rozkaz #include "nazwa_pliku" jest podobny, z tym, że plik jest szukany w katalogu, w którym znajduje się kod źródłowy programu. Np.

#include "deklaracje.h"

Rozkazy #include powinny się znajdować zawsze na początku programu, jako pierwsze. Rozszerzenie '.h' dodawane do nazwy pliku jest standardowym określeniem, że chodzi o plik nagłówkowy.

Dyrektywa #define pozwala na utworzenie własnego odpowiednika jakiegoś ciągu znaków. Jeżeli podczas kompilacji preprocesor znajdzie taki tekst w programie, zastąpi go podanym przez nas ciągiem znaków. Np.

#define ROZMIAR_X 10

Jeżeli w pisanym przez nas programie znajduje się tablica, która w wielu miejscach programu przeszukujemy, musimy podawać jej wielkość. Jeżeli będziemy używali po prostu liczby, zmiana rozmiaru tablicy będzie wymagała szeregu zmian w programie. Łatwiej zdefiniować własne makro, które odpowiada rozmiarowi tablicy i podczas procedur przeszukiwania używać tego właśnie makra. Zmiana rozmiaru tablicy sprowadzi się wtedy jedynie do modyfikacji definicji makra.

#undef pozwala preprocesorowi na usunięcie podanego makra, np.

#undef ROZMIAR_X

Rozkazy #if, #ifdef, #ifndef, #else, #endif służą do przeprowadzenia kompilacji warunkowej. Każde #if musi być zakończone przez #endif. Użycie #else jest opcjonalne. Rozkazy #ifdef i #ifndef oznaczają kolejno: jeśli jest zdefiniowane, jeśli nie jest zdefiniowane. Najlepiej zobaczyć, jak działają instrukcje kompilacji warunkowej poprzez prześledzenie zawartości plików nagłówkowych.

Warto zauważyć, że po dyrektywach preprocesora nie piszemy średników.

Tym samym dobrnęliśmy do końca podstaw języka C. Zdaję sobie sprawę, że nie wszystkie rzeczy zostały omówione.

Biblią programisty C jest książka Briana W. Kernighana i Dennisa M. Ritchie'go "Język C". Zawiera ona kompletną definicję języka i jest po prostu książką, a więc czymś, co się da przeczytać i zrozumieć.

Co jednak z C na Amigę? Otóż właściwie to nie ma „języka C na Amigę". Cała zabawa w programowanie w C na Amidze polega na znajomości systemu operacyjnego Amigi i umiejętnym wykorzystaniu procedur, znajdujących się w ROMie i w bibliotekach dyskowych. Sześć części kursu pozwoliło na przedstawienie, o co w C chodzi. O co chodzi w Amidze, o tym opowie - być może - następny kurs.

Jarosław Chrostowski